Plataforma para el fomento turístico basado en recompensas mediante tecnologías Blockchain

Sergio Pino Holgado

Trabajo de fin de grado del Grado en Ingeniería de Software

Departamento de Ingeniería del Software e Inteligencia Artificial
Facultad de Informática
Universidad Complutense de Madrid

Septiembre 2019

Director: Juan Antonio Recio García
Colaborador: Felipe Santi (Sismotur)

Agradecimientos

Quiero emplear estas primeras lineas para agradecer en especial a mis padres Andrés y Marylen, a mi hermano Alex y a mi futura esposa Marta, toda esa paciencia destilada capáz de soportarme en las situaciones más difíciles por las que he pasado durante la carrera ya que sin ellos nada de esto hubiera sido posible, estoy seguro.

También quiero dar las gracias a Juan Antonio, profesor y tutor por haberme acompañado y guiado durante estos meses de trabajo.

Y a Felipe, por haberme mostrado el lado más profesional de la temática del proyecto y haberme concedido parte de su experiencia en el mundo real.

Índice general

Agradecimientos **3**

Lista de figuras **11**

Lista de tablas **12**

1 Introducción **15**

1.1 Resumen 15

1.2 Abstract 17

1.3 Estructura 18

2 Motivación y objetivos **19**

2.1 Motivación 19

2.2 Objetivos 21

3 Estado del arte **23**

3.1 Dominio 23

3.1.1 Sismotur 23

3.1.2 Inventrip 24

3.2 Sistemas turísticos 27

3.3 Beacons o puntos de señalización 28

3.4 Blockchain 29

3.4.1 ¿Qué es Blockchain? 29

3.4.2 ¿Cómo funciona Blockchain? 29

3.4.3 ¿Quién mantiene las redes Blockchain? 30

3.5 Bitcoin 31

3.5.1 ¿Cómo funciona Bitcoin? . . . 31
3.5.2 ¿Qué es la "minería"? . . . 32
3.6 Stellar . . . 33
3.6.1 ¿Cómo funciona Stellar? . . . 33
3.7 Ethereum . . . 34
3.7.1 ¿Cómo funciona Ethereum? . . . 34
3.8 Elección del sistema adecuado . . . 35

4 Proceso de desarrollo **37**
4.1 Herramientas . . . 37
4.1.1 Productividad y comunicación . . . 37
4.1.2 Código . . . 38
4.1.3 Desarrollo . . . 38
4.2 Metodología . . . 39
4.2.1 Agile . . . 39

5 Arquitectura **41**
5.1 Tecnologías . . . 42
NodeJS . . . 42
JavaScript . . . 42
HTML5+CSS3 . . . 42
MongoDB . . . 43
5.2 Arquitectura del sistema . . . 44
5.2.1 Webapp . . . 45
5.2.2 API . . . 46
5.2.3 Almacén de datos . . . 47
5.2.4 Stellar Network . . . 47
Uso de Stellar Network en la aplicación . . . 48

6 Casos de uso **55**
6.1 Actores del sistema . . . 56
6.2 Generales . . . 57

6.2.1 Crear cuenta . . . 57
6.2.2 Acceso al sistema . . . 57
6.2.3 Cerrar sesión . . . 58
6.3 Turista . . . 59
6.3.1 Suscribirse a una ruta . . . 59
6.3.2 Validar punto de prueba de paso . . . 59
6.3.3 Validar recompensa . . . 60
6.4 Hostelero . . . 64
6.4.1 Crear nuevo punto de recompensa . . . 64
6.4.2 Obtener listado de puntos de recompensa . . . 64
6.4.3 Borrar punto de recompensa . . . 64
6.5 Institución . . . 66
6.5.1 Crear un nuevo punto de prueba de paso . . . 66
6.5.2 Obtener listado de puntos de prueba de paso . . . 66
6.5.3 Borrar punto de prueba de paso . . . 66

7 Conclusiones **69**
7.1 Resumen . . . 69
7.2 Competencias adquiridas . . . 71
7.3 Trabajo futuro . . . 72
7.4 Summary . . . 73
7.5 Knowledge acquired . . . 74
7.6 Future work . . . 75

8 Apéndices **77**
8.1 Requisitos funcionales . . . 77
8.1.1 Generales . . . 77
8.1.2 Turista . . . 80
8.1.3 Hostelero . . . 83
8.1.4 Institución . . . 88
8.2 Repositorios . . . 95
8.3 Como desplegar . . . 96

8.3.1 Preparación del entorno 96
Almacén de datos 96
API 96
WebApp 96
8.4 API endpoints 97
8.4.1 Cuentas 97
8.4.2 Autenticación 98
8.4.3 Puntos de prueba de paso 98
8.4.4 Puntos de recompensa 99
8.4.5 Recompensas 100

9 Referencias **103**

Glosario

API Application Programming Interface. Se trata de una capa de software mediante la cual se expone un servicio o servicios a sus posibles consumidores. De esta forma, el usuario realiza peticiones y obtiene respuestas elaboradas para ellas sin conocer cual es el funcionamiento del o de los servicios.. 5, 7, 18, 44, 45, 47, 48, 71, 72, 74, 75, 95, 96, 97, 98, 99, 100, 101, 102

Bluetooth LE Bluetooth Low Energy es una tecnología inalámbrica de bajo consumo que sirve para conectar uno o varios dispositivos cercanos entre sí.. 24, 28

Cadena de bloques Blockchain. Es la parte central sobre la que se fundamenta Bitcoin en particular y muchas criptomonedas en particular. Es aquí donde todas las transacciones que se llevan a cabo son almacenadas.. 29

CSS Cascading Style Sheet. Es la forma de escribir una serie de reglas ordenadas y previamente definidas que nos permiten determinar como se verán los elementos gráficos de nuestra aplicación, generalmente web pues existen otros lenguajes de programación como JavaFX que también permiten su uso en aplicaciones de escritorio.. 5, 42

Endpoint La forma en la que una API expone sus capacidades se determina mediante los endpoints. Un endpoint estara definido por una URL y el metodo HTTP que acepta (GET, POST, PUT, DELETE...); aunque en ciertas definiciones tambien se suele incluir los datos que espera recibir así como su forma y un ejemplo de los datos que devolverá.. 7, 18, 45, 48, 97, 98, 99, 100, 101, 102

HTML Hypertext Markup Language. Es un lenguaje de marcado y no un lenguaje de programación como se suele confundir. Se utiliza para maquetar las páginas que conforman un sitio web. Esta maquetación se hace mediante el uso de unas etiquetas ya definidas que son empleadas por el usuario para indicar como se debe distribuir y mostrar el contenido al usuario.. 5, 42

HTTP Hyper Text Transfer Protocol. Es posiblemente el protocolo mas conocido de internet, ya que es el empleado en la transmisión de la información en Internet. Establece las reglas para que un sistema solicitante sea capaz de realizar las peticiones adecuadas y obtener la información requerida de otro sistema informático.. 45

JavaScript Es un lenguaje de programación que nació originalmente para ser usado en la web.. 5, 42, 71, 74

MongoDB Es un sistema gestor de bases de datos abierto orientado a documentos, por lo que no almacena la información de forma relacional en tablas si no en documentos JSON.. 5, 42, 47

NodeJS Es un entorno en tiempo de ejecución que permite la ejecución de código Javascript fuera de un navegador.. 5, 38, 42

Peer-to-peer Una red peer-to-peer o red entre iguales es un tipo de red en la que todos los nodos que se conectan a ella son tratados como iguales. No existe un servidor central y cualquier nodo puede actuar indistintamente como cliente o servidor.. 30

Índice de figuras

3.1 Planificación de viaje con Inventrip 25
3.2 Integración de un Beacon en señal turistica 25

5.1 Arquitectura del sistema 44

6.1 Creación de una cuenta 57
6.2 Acceso al sistema 58
6.3 Cerrar la sesión 58
6.4 Suscribirse a una ruta 59
6.5 Validar punto de prueba de paso 60
6.6 Validar recompensa 61
6.7 Crear punto de recompensa 64
6.8 Listado puntos de recompensa 65
6.9 Borrar punto de recompensa 65
6.10 Crear un punto de prueba de paso 66
6.11 Obtener listado de puntos de prueba de paso 67
6.12 Borrar punto de prueba de paso 67

Índice de cuadros

8.1 Crear cuenta . . . 77
8.2 Acceder al sistema . . . 78
8.3 Cerrar sesión . . . 79
8.4 Consultar rutas disponibles . . . 80
8.5 Suscribirse a una ruta . . . 80
8.6 Validar punto de prueba de paso . . . 80
8.7 Consultar puntos de prueba de paso . . . 81
8.8 Consultar puntos de recompensa . . . 82
8.9 Consultar cartera . . . 82
8.10 Validar recompensa . . . 83
8.11 Consultar rutas disponibles . . . 83
8.12 Consultar puntos de prueba de paso . . . 84
8.13 Crear punto de recompensa . . . 84
8.14 Consultar puntos de recompensa . . . 85
8.15 Actualizar punto de recompensa . . . 85
8.16 Borrar punto de recompensa . . . 86
8.17 Validar recompensa de un turista . . . 87
8.18 Crear ruta . . . 88
8.19 Consultar rutas . . . 89
8.20 Actualizar ruta . . . 89
8.21 Borrar ruta . . . 90
8.22 Crear punto de prueba de paso . . . 91
8.23 Consultar puntos de prueba de paso . . . 92
8.24 Actualizar punto de prueba de paso . . . 92

8.25 Borrar punto de prueba de paso 93

Capítulo 1

Introducción

1.1. Resumen

España tiene una fuerte cultura y gran diversidad de tradiciones, no solo a nivel nacional si no autonómico. Su posición geográfica permite disfrutar de un clima bastante regular, sobre todo en el sur. La gastronomía también tiene un fuerte marcado regional, ofreciendo un abanico de posibilidades muy extenso y de calidad. Es por eso por lo que España recibe un total aproximado de 82 millones de turistas extranjeros al año, recibiendo más turistas que Estados Unidos y solo por debajo de Francia.

Con esta afluencia de gente se ha tenido que invertir mucho para satisfacer las necesidades del turista, por eso España tiene una de las mejores infraestructuras de servicios turísticos a nivel mundial. Con las nuevas tecnologías que se han ido desarrollando durante las ultimas décadas el turista goza de servicios de primer nivel. Hoy en día existen numerosas aplicaciones o paginas web donde encontrar hoteles, vuelos, trenes, restaurantes y actividades que hacer en cualquier ciudad del mundo. Pero se esta perdiendo el turismo cultural o histórico, sobre todo en la gente joven.

Sismotur es una empresa turística, líder en la elaboración de planes de señalización y en servicios de información turística inteligente. Inventrip, una de sus plataformas, dispone la

oferta turística de un lugar. Mediante el uso de beacons, una tecnología basada en Bluetooth, permite recibir directamente en un dispositivo móvil la información relevante de ese lugar; estas aplicaciones están incorporando recompensas a los usuarios y es necesario un mecanismo que garantize y gestiona las acciones de los usuarios. En este Trabajo de Fin de Grado se ha colaborado con Sismotur para crear una aplicación que junto a su red de señalización beacon, los dispositivos móviles de los usuarios y la plataforma de criptomoneda Stellar, permita incentivar a los turistas a informarse mas y mejor sobre los lugares que visitan. Ofreciendo recompensas en forma de criptomoneda a los turistas que realicen las rutas propuestas.

1.2. Abstract

Spain has a strong culture and a great diversity of traditions, not only nationally but also autonomously. Its geographical position allows to enjoy a fairly regular climate, especially in the south. Gastronomy also has a strong regional mark, offering a wide range of possibilities and quality. That is why Spain receives a total of approximately 82 million foreign tourists a year, receiving more tourists than the United States and only below France.

With this influx of people has had to invest a lot to meet the needs of tourists, so Spain has one of the best infrastructures of tourism services worldwide. With the new technologies that have been developed during the last decades the tourist enjoys services that 20 years ago could not even imagine. Today many of us know applications or web pages where to find hotels, flights, trains, restaurants and activities to do any city in the world. But cultural or historical tourism is being lost, especially in young people. It is very common to see people posing in front of the Puerta de Alcalá, the Eiffel Tower or the Fontana di Trevi in social networks, they have gone to any of these cities, visit them, enjoy them, but they do not really know very well his story.

Sismotur is a tourism company, leader in the development of signal plans and intelligent tourist information services. Inventrip, one of its platforms, puts at your disposal the tourist offer of a place. Through the use of beacons, a technology based on Bluetooth, allows you to receive directly on your mobile device the relevant information of that place. In this End of Degree Project, we have collaborated with Sismotur to create an application that, together with its beacon signaling network, the users' mobile devices and the Stellar cryptocurrency platform, allows tourists to learn more and better about the places they visit. Offering rewards in the form of a cryptocurrency to tourists who make the proposed routes.

1.3. Estructura

El contenido del documento sigue la siguiente estructura:

- Breve resumen de este Trabajo de Fin de Grado junto a la estructura del mismo.
- En el segundo capítulo se describen la motivación para el desarrollo del trabajo y los objetivos perseguidos.
- En el tercer capítulo se explica que son los sistemas turísticos, los beacons y la tecnología Blockchain: los diferentes tipos de criptomonedas que se han barajado para el desarrollo del trabajo.
- En el cuarto capitulo se detalla como ha sido la organización a nivel interno, las herramientas que se han utilizado para ello y las que se han empleado para desarrollar el software. También se explica que tipo de metodología se ha seguido y los motivos.
- En el quinto capítulo se detalla tanto la arquitectura utilizada como las tecnologías empleadas para el desarrollo de la aplicación.
- En el sexto capitulo se describen los casos de uso de todas las acciones que pueden llevar a cabo los usuarios. Las precondiciones, postcondiciones y los errores que podrían darse están vinculados en el apéndice.
- En el séptimo capítulo se encuentra un resumen de lo que ha sido el trabajo, lo que ha supuesto y el posible trabajo futuro.
- En el ultimo capítulo se describe como desplegar y preparar el entorno para su ejecución. También detallamos los endpoints de la API y la descripción funcional.

Capítulo 2

Motivación y objetivos

2.1. Motivación

Una de las tecnologías mas recientes y que no están incluidas en el plan de estudios es Blockchain; se eligió como tema principal para la realización de este trabajo para tener un primer contacto y poder realizar un prototipo de implantación real.

Sismotur es una empresa dedicada al turismo que ha apostado desde sus inicios por la tecnología. Sus dos grandes pilares de negocio actualmente son sistemas para la gestion turística de forma completamente informatizada. Sismotur se encarga de identificar problemas reales y diseñar soluciones a medida aplicando métodos que involucren a las nuevas tecnologías.

Uno de los actuales problemas en el sector turístico es la dificultad de manejar a las grandes masas de turistas, que en muchas ocasiones llegan a colapsar ciertas partes de los lugares que visitan. Sismotur supo detectar este problema y diseñaron una solución que en conjunto con los ayuntamientos con los que trabajan, pretende mostrar a los turistas nuevas formas de visitar esas ciudades o pueblos y les permite igualmente planificar sus viajes mediante herramientas sencillas.

Con Sismotur se buscó una forma de acelerar ese proceso incorporando para ello el uso de alguna tecnología Blockchain. Una buena forma de conseguir que los turistas entrasen en esta

modalidad de hacer turismo es recompensarles de alguna forma. Algunas tecnologías Blockchain permiten realizar pagos e incorporarlos de forma sencilla a una aplicación mediante código, abstrayendo por completo de la responsabilidad de gestionar estos pagos a los desarrolladores o instituciones que implanten este sistema.

A pesar de que Sismotur dispone ya de herramientas para la gestión turística, no disponen aún de un sistema que permita guiar a los turistas por rutas predefinidas, compruebe que las han realizado y les recompense al termino de estas, todo de forma automatizada. También es interesante poder delegar la gestión de los pagos en un sistema seguro, descentralizado y fiable para poder centrarse en lo realmente importante que es el turismo en sí.

2.2. Objetivos

Los objetivos principales de este trabajo son los siguientes:

- Crear una herramienta con la capacidad necesaria para, en conjunto con las herramientas de las que ya dispone Sismotur, dar la posibilidad a los servicios turísticos de recompensar a los usuarios por realizar determinadas rutas ya predefinidas y así conseguir repartir a los turistas y evitar que se aglutinen en los lugares más concurridos, ampliando la oferta al crear mas alternativas y creando experiencias mas dinámicas para los usuarios. Estas recompensas son remuneraciones mediante el pago de ciertas cantidades de una criptomoneda.
- Aplicar los conocimientos y tecnologías adquiridas durante el grado.
- Aprender como funcionan las nuevas tecnologías basadas en Blockchain, algo realmente novedoso y que ha llegado para cambiar el mundo tecnologíco tal y como lo conocemos hoy.
- Definir e implementar una solución a un problema real y ser capáz de generar un prototipo funcional que haga uso de las nuevas tecnologías.
- Tener la posibilidad de trabajar en conjunto con una empresa y conocer el proceso de desarrollo de software en un entorno de trabajo real.

Capítulo 3

Estado del arte

3.1. Dominio

3.1.1. Sismotur

Sismotur es una empresa orientada y dedicada al sector del turismo, fundada en el año 2000 con la finalidad de ofrecer consultoría enfocada en la creación e implantación de sistemas de señalizacion inteligentes.

Para ello se han servido de las ultimas tecnologías del mercado ofreciendo dos herramientas muy potentes, así es como ellos mismos las definen:

> **"Inventrip"** es su principal servicio, que pone a disposición de los usuarios un completo sistema a modo de panel de control desde el que se pueden gestionar viajes teniendo acceso a diversa información turística del lugar a visitar, así como hacer las reservas en los establecimientos, gestionar rutas a seguir, y como no compartir todos estos datos en redes sociales.
>
> **"Signing"** para que Inventrip pueda funcionar es necesario que todos los lugares en donde funciona se haya preparado el entorno para que el usuario pueda interactuar

con el, y aquí entra Signing, que es la plataforma para la elaboración de planes de señalización que propone Sismotur, que es un servicio web desde el que gestionar y planificar la señalización de un pueblo, ciudad...

Hoy en día el perfil del turista ha cambiado mucho, gracias a las nuevas tecnologías que tenemos al alcance de la mano podemos conocer cualquier cosa en cualquier momento y lugar. Sismotur aprovecha estas tecnologías para junto a administraciones publicas ofrecer un servicio turístico de calidad, veraz y unificado en un solo sistema. Ya que las búsquedas en google o motores de búsqueda de turismo ofrecen información fragmentada y que podría no estar verificada por alguien con el conocimiento adecuado sobre el lugar en cuestión.

Este sistema acompaña al turista desde que empieza a planificar el viaje en su casa hasta que una vez finalizado ofrece su valoración. Ofreciendo en todo momento un servicio exclusivamente pensado en la satisfacción del turista.

3.1.2. Inventrip

El objetivo final de este Trabajo de Fin de Grado es que fuera implementado en la aplicación Inventrip, por eso creo que es importante ahondar un poco más en su funcionamiento y las posibilidades que ofrece.

Inventrip pone a tu disposición la oferta turística de un lugar, con ella puedes planificar viajes a medida y disponer de tu planificación en cualquier dispositivo.

En tres sencillos pasos te permite crear una agenda de viaje, seleccionando por días lo que se quiere visitar, donde comer y donde dormir. Una vez finalizada la planificación puedes compartirla con tus compañeros de viaje.

Para ello Sismotur cuenta con dos canales de información, el primero es la señalización turística que implantan en los destinos. Las administraciones publicas sacan a concurso la administración e implantación de las señales físicas para informar y guiar al turista en un destino. Además de eso Sismotur integra en sus señales unos dispositivos llamados Beacons, que utilizan una señal Bluetooth LE que ofrece información directa sobre el lugar que se esta visitando.

Figura 3.1: Planificación de viaje con Inventrip

Estos dispositivos se integran en la propia señal de información como se puede apreciar en la siguiente ilustración.

Figura 3.2: Integración de un Beacon en señal turistica

El segundo canal de información seria la propia aplicación de Inventrip, que dispondría de toda la información previamente explicada de forma online.

De esta forma cuando el turista esta usando la App de Inventrip en su dispositivo móvil cerca de una señal que integra un Beacon estos dos dispositivos se comunican juntando toda la información en un mismo lugar.

Esto le permite al turista obtener respuesta a preguntas comunes como ¿Dónde estamos?, ¿Qué tenemos alrededor?, ¿Qué estamos visitando? Como si te estuviera atendiendo un agente de la oficina de turismo «in situ».

Este tipo de señalización ofrece ventajas notables, por ejemplo, puedes actualizar la información que ofrece un Beacon de forma muy simple y añadir todos los idiomas que quieras. Algo que en una señal física convencional no puedes hacer, ya que tienes un espacio limitado. También puedes acceder a toda la información que brindan los Beacon sin tener conexión a internet en tu dispositivo móvil, es común que cuando viajas a menudo tengas que desactivar la conexión a internet para evitar elevados costes de roaming.

3.2. Sistemas turísticos

Entendemos en este ámbito un sistema como un conjunto ordenado de herramientas que trabajan de forma sinérgica para facilitar las labores de todos los actores que están relacionados con él; en este caso concreto al tratarse de un sistema turístico los actores a grandes rasgos, son las personas que acuden a los lugares turísticos (demanda) y por otra parte las personas que trabajan en los establecimientos prestando los servicios para los turistas (oferta). Así, podemos dividir estos actores en dos grandes grupos:

- **Demanda turística.** Es el conjunto de productos y servicios que los turistas esperan obtener cuando acuden a un destino.
- **Oferta turística.** Por el contrario, es el conjunto de esos productos y servicios que se ponen a disposición de los turistas para su uso y disfrute.

Es claro que ambos conceptos van de la mano pues un lugar con mucha demanda que apenas tiene que ofrecer a las personas que acuden a él no tendrá el mismo éxito que otro con mayor oferta. Es por ello que se trata de buscar siempre un equilibrio lógico entre estos dos factores para que los turistas puedan disfrutar con lo que realmente esperan obtener de un lugar y por otra parte, que la oferta no sea excesiva.

Sismotur, como empresa dedicada al sector turístico, dispone actualmente de dos grandes herramientas para poder gestionar esta situación. Por un lado esta «Signing», que es empleado para la gestión de planes de señalización, permitiendo diseñarlos e implantarlos. Y por otro lado «Inventrip» que junto con la anterior herramienta, permite hacer uso de esas señales para ofrecer información turística y crear viajes a medida.

3.3. Beacons o puntos de señalización

La forma que utiliza Sismotur para crear sus puntos de señalización físicos se llama «Beacon»; se trata de unos dispositivos hardware formados por una coraza resistente anti-vandálica que oculta un dispositivo Bluetooth LE en su interior. Este dispositivo se dedica a emitir señales con mensajes o avisos que son recibidos por dispositivos que se encuentren cerca esperando a recibir dicha información.

Gracias a su reducido tamaño es posible instalarlos en prácticamente cualquier sitio, aunque normalmente están en puntos de información turística, paneles informativos, etc.

Su uso principal es para proporcionar ciertos datos en determinadas situaciones; al estar integrados en estas estructuras y ser dispositivos que únicamente envian información no son manipulables por terceras personas, algo de vital importancia porque con nuestra aplicación pretendemos emplear estos puntos como prueba de paso, pudiendo verificar que una persona ha estado físicamente en cierto punto.

3.4. Blockchain

Una de las partes más importantes de este trabajo ha sido la investigación realizada para poder implementar con éxito un sistema de remuneraciones usando criptomonedas. La pieza fundamental sobre la que se han construido la mayoria de ellas es Cadena de bloques o «cadena de bloques» en castellano.

3.4.1. ¿Qué es Blockchain?

Creo que la forma más fácil para dar a entender que es una Cadena de bloques es recurriendo al ejemplo típico que se emplea en este caso: es como un gran libro contable donde todas las transacciones que son realizadas por todo el mundo son registradas (con mayor o menor información) de forma ordenada segun el tiempo en que son realizadas; todas estas transacciones son revisadas por millones de máquinas a lo largo de todo el mundo, lo que la hace descentralizada y muy segura ya que evita que alguien con ideas destructivas sea capaz de manipularla.

Se suele utilizar este ejemplo porque el uso de una cadena de bloques se popularizó con Bitcoin, que sigue siendo a día de hoy la criptomoneda más famosa y la primera de todas, o al menos ha sido el primer concepto de moneda electrónica que ha tenido implantación real y no se ha quedado en una mera prueba de concepto.

3.4.2. ¿Cómo funciona Blockchain?

Una cadena de bloques como su nombre indica, está formada por bloques encadenados, esto quiere decir que una vez que una transacción entra dentro de la cadena, lo hace simbólicamente en forma de «bloque» y queda vinculado de forma inseparable al que fué el último bloque antes que él. De esta forma se garantiza la trazabilidad de todas y cada una de sus transacciones.

Otro de los puntos más destacables cuando se habla de una cadena de bloques es la forma en que su información es mostrada a los usuarios, pues cualquiera puede conectarse a la red y obtener este «libro contable» y tener acceso a todas las transacciones que se han realizado desde

el principio de los tiempos (o génesis como se le llama en Bitcoin).

Cuando se realiza una transacción esta es incorporada a la cadena de bloques de forma temporal, ya que el resto de máquinas deben «confirmarla» antes de ser válida para el resto de usuarios. De esta forma, cuando un bloque es incorporado a la cadena, primero debe ser confirmada por un cierto número de máquinas antes de que el resto de usuarios lo tomen como válido. Cada confirmación que va obteniendo un bloque lo hace más robusto.

3.4.3. ¿Quién mantiene las redes Blockchain?

Al tratarse de una red Peer-to-peer son los propios usuarios los que la mantienen. De esta forma, todos los nodos conectados a la red contribuyen realizando cargas de trabajo; si un nodo de la red cae o desaparece (el usuario decide no continuar coenctado a ella), otro nodo tomará el relevo y realizará el trabajo.

3.5. Bitcoin

Cuando se habla de Bitcoin se suele relacionar con el concepto de criptomoneda, pero Bitcoin es mucho más que eso. Efectivamente, el proyecto Bitcoin fué concebido como un nuevo sistema de pago y moneda electrónica a la vez; pero lo cierto es que a día de hoy Bitcoin se ha convertido en una referencia, no sólo en el mundo de las criptomonedas si no en internet en general y esto es debido a su forma de funcionar ya que los usuarios han ido descubriendo nuevas formas de uso para su cadena de bloques.

De forma muy resumida podriamos decir que Bitcoin se trata de un sistema electrónico de pagos a través de internet, seguro, rápido y sin prácticamente comisiones.

Para entender un poco más lo importante de este proyecto hay que saber que su autor es completamente desconocido, lo único que se sabe es que el borrador que fué publicado con la especificación de Bitcoin en el año 2009 lo firmaba un tal Satoshi Nakamoto. Por supuesto todo el código de Bitcoin es abierto con lo que cualquier persona con los conocimientos adecuados puede realizar modificaciones en su código. Esto convierte a Bitcoin en algo de todos y esa es a mi modo de ver, su mayor fortaleza.

3.5.1. ¿Cómo funciona Bitcoin?

El funcionamiento de Bitcoin es sencillo en cuanto a su uso pero complejo estructuralmente. En rasgos generales, un usuario que quiera conectarse a la red de Bitcoin necesitará un cliente, que es una aplicación para facilitar la creación y gestión de las transacciones. Existen dos tipos de cliente: los ligeros y los pesados. A diferencia de los primeros que no necesitan hacer nada antes de usarlos, los clientes pesados requieren de la descarga previa del libro de cuentas de Bitcoin, donde están todas y cada una de las transacciones desde el inicio de la Blockchain hasta el momento actual de la descarga; esto le permite al cliente trabajar con seguridad e independencia pues no depende de otros clientes para realizar elaborar una transacción válida.

Por otra parte tenemos la parte no visible para los usuarios finales de Bitcoin, que es todo lo que esta por debajo que hace funcionar a la red y que las transacciones sean posibles. Cada

transacción es verificada por cientos o miles de otros nodos, lo que garantiza la descentralización y la seguridad de que una vez sea finalmente incluída en el libro contable se tiene la certeza de que esa transacción en efecto a sido realizada y que todos los datos que en ella se incluyen son completamente válidos.

3.5.2. ¿Qué es la "minería"?

Esta es quizá, la parte mas compleja y estudiada de Bitcoin. Es la forma en la que cualquier usuario que tenga el hardware necesario capáz de generar la potencia de procesado requerida puede procesar transacciones y obtener bitcoins por el esfuerzo realizado. Procesar una transacción se refiere al acto de confirmar transacciones que están pendientes de ser incorporadas al libro contable. Otro aspecto a destacar de este proceso es que en el proceso se incluye un paso muy aleatorio, lo que garantiza que no se podrá añadir una transacción fácilmente por cualquier persona y mantener así el libro seguro pues nadie es capaz de controlar a su antojo las transacciones, pudiendo borrar o añadir fondos a placer.

3.6. Stellar

Tras la «estela» de Bitcoin nacieron multitud de proyectos similares que pretenden mejorar las carencias que presenta Bitcoin. Stellar es un proyecto de código abierto pensado para realizar transferencias entre dinero real o fiat y criptomonedas. Stellar también tiene su propia moneda electrónica, llamada Stellar Lumens (XLM) que es usada en dichas transferencias de dinero. La gran ventaja que propone Stellar sobre otras es la posibilidad de poder realizar las transferencias sin importar origen y destino ni físico ni del tipo de moneda. El proyecto surgió de la necesidad de crear un sistema más rápido, más barato y más confiable de lo que conocemos hoy en día.

3.6.1. ¿Cómo funciona Stellar?

De forma similar a Bitcoin, en la red de Stellar se gestionan todas las transacciones de forma completamente descentralizada, no existe ningún servidor central, toda su información se encuentra distribuida a lo largo del globo en millones de nodos conectados entre sí mediante internet.

Cualquier usuario con acceso a la red de redes puede conectarse a Stellar y realizar transacciones dentro de esta. Para poder acceder a la red de Stellar se hace mediante una herramienta propia llamada «Horizon» que nos permite la conexión con uno de los Stellar Cores que forman la llamada «columna vertebral de Stellar».

Los Stellar Cores se encargan, entre otra muchas cosas, de garantizar que la red siga funcionando de forma descentralizada y de poner cierto orden para evitar duplicidad de bloques, hacer el seguimiento de las transacciones, etc. Cada uno de estos Stellar Cores es mantenido por diferentes agrupaciones de personas y/o empresas.

Por otra parte tenemos la Stellar Network, que no es otra cosa que la agrupación mundial de los Stellar Cores.

3.7. Ethereum

A diferencia de otros proyectos como Bitcoin o Stellar, Ethereum es una plataforma y no solo una criptomoneda, aunque también tiene su propia divisa electrónica que son los Ether (ETH). La principal utilidad de Ethereum, o para lo que fué concebida es para la creación y gestión de Smart Contracts o contratos inteligentes. Este proyecto fué creado en origen por Vitalik Buterin.

3.7.1. ¿Cómo funciona Ethereum?

Al igual que Bitcoin y Stellar, Ethereum también mantiene su propia Blockchain donde va almacenando todas las transacciones que se realizan. La principal característica que hace diferente a Ethereum de otras criptomonedas es que dispone de un lenguaje de programación propio que permite crear Smart Contracts (contratos inteligentes).

Estos contratos son programados e incorporados a la blockchain. Una vez que las condiciones de ese contrato se cumplen, el código es ejecutado. Esto permite realizar transacciones seguras prescindiendo de la persona que haría de intermediario y que podría ser manipulada o simplemente imparcial.

Al incluirse en la blockchain estos contratos se cumplen pase lo que pase y justo como fueron escritos.

3.8. Elección del sistema adecuado

Habiendo barajado la posibilidad de usar Bitcoin, por ser la más famosa o Ethereum por ser la segunda más conocida, finalmente se decidió realizar el trabajo con Stellar por diversos motivos, pero los más destacados fueron:

- Lo primero de todo la ingente cantidad de documentación y herramientas que pone Stellar a disposición de los desarrolladores para la implantación de nuevas herramientas.
- El precio a pagar por cada transacción es prácticamente inapreciable en comparación con otras criptomonedas.
- No se hace uso de la minería, como en Bitcoin, lo que facilita muchas tareas sin sacrificar seguridad.

La parte negativa de Stellar, aunque todo son rumores por el momento es que aún no está totalmente descentralizado ya que gran cantidad de sus tokens siguen perteneciendo a la fundación.

Capítulo 4

Proceso de desarrollo

4.1. Herramientas

En esta sección se detallan cuales han sido las herramientas que empleadas a lo largo de todo el proceso.

4.1.1. Productividad y comunicación

Se ha optado por mantener una comunicación casi exclusivamente online y para ello nos ayudamos de algunas de las herramientas disponibles:

- Google Hangouts: con la que se realizaban las videollamadas para concretar aspectos que nos era más complicado hablar por escrito.
- Telegram: la gran mayoria de las comunicaciones eran mediante esta vía, lo que ha facilitado enormemente las tareas pues en cualquier momento podíamos solicitar ayuda sin tener en cuenta la hora y/o horarios de la otra persona y ser contestados a la máxima brevedad.
- Google Drive: ha permitido compartir diversa información que por temas de organización era más cómodo que enviarla por Telegram. Al estar disponible toda esta información en

único punto es más sencillo consultar cualquier tema.

4.1.2. Código

Toda la gestión tanto de código como de documentación (o la propia memoria) ha sido realizada integramente usando GitHub, concretamente sus repositorios privados gratuitos. Toda esta gestión se hace mediante el uso de la herramienta Git, bien desde la terminal de comandos o desde el cliente de escritorio que provee GitHub o el de Atlassian que se llama SourceTree. Esta forma de organizar y gestionar el código ha permitido el código sincronizados en todo momento de forma eficiente y segura, pues funciona a la vez como una copia de seguridad, al no tener el código centralizado en un único punto.

Otra de las bondades de Git es que permite la creación de «ramas» o «branches»; estas ramas permiten bifurcar el código generando una versión paralela del mismo y cuyos cambios no afectan a la rama principal o «master». Esto ha facilitado el flujo de trabajo.

4.1.3. Desarrollo

Para el desarrollo se han empleado diversas herramientas. Cada uno tenía sus propias preferencias y no ha sido un problema gracias al uso de Git como gestor de código.

- WebStorm: IDE empleado en la elaboración de las dos aplicaciones escritas en NodeJS; permite depurar el código de forma avanzada para poder corregir errores o mejorar el rendimiento y la estabilidad.
- Visual Studio Code: es un editor de texto «hipervitamido» que tiene la posibilidad de agregar plugins de terceros ampliando su abanico de posibilidades de forma prácticamente ilimitada.
- TexMaker: entorno para la elaboración de documentos usando LaTeX.

4.2. Metodología

El primer paso antes de comenzar a trabajar en cualquiera de las partes del proceso fue definir la metodología a emplear, siendo Agile la que más se ajustaba a las necesidades del proyecto, pues uno de los problemas a resolver era la distancia y el horario de los participantes del trabajo. De esta forma se decidió planificar reuniones semanales y conversaciones diarias en las que resolver los problemas que fueran surgiendo.

4.2.1. Agile

Se ha dividido el trabajo en periodos semanales (iteraciones) en los que se marcaban unos objetivos a cumplir. Cada domingo de la semana se realizaba una revisión del estado del proyecto para determinar en qué punto estaba y donde se pretendía llegar en la próxima iteración.

Cada semana se asignaban las tareas que debían ser terminadas en el período establecido. No hubo ningún inconveniente con esta forma de trabajo pues había responsabilidad y compromiso suficiente para que cada tarea fuera completada en el tiempo previsto y no pudiera bloquear otras tareas. Para garantizar que todas fueran terminadas en el tiempo establecido las tareas eran partes pequeñas y sencillas de ejecutar. Cuando existía un bloqueo se compartía el problema y se daba una solución en común.

En forma de «daylies», cada día había una conversación para comprobar el trabajo realizado el día anterior y comprobar posibles bloqueos o dudas. Por las dificultades que suponía la distancia, estas reuniones se realizaban usando Telegram y/o Google Hangouts.

Capítulo 5

Arquitectura

En esta sección se describe como está formado el sistema y por qué. Se detallan las tecnologías que han sido empleadas, los patrones de diseño y la estructura de cada una de las piezas. Durante todo el proceso se han tenido en cuenta los estándares de desarrollo, así como las convenciones de cada tecnología, aunque en algunos casos haya sido necesario idear una forma alternativa bien por ausencia de estándares o bien por necesidad de adaptar algo muy concreto al proyecto.

5.1. Tecnologías

A continuación se listan algunas de las tecnologías más importantes que se han empleado para la construcción del sistema así como una breve explicación (si la hubiera) de porqué se han elegido sobre otras disponibles.

NodeJS

Puesto que uno de los requisitos era el emplear JavaScript como lenguaje principal del proyecto, NodeJS era el claro ganador frente a sus posibles rivales puesto que no existía la necesidad de formarnos pues ya disponíamos de los conocimientos necesarios.

JavaScript

Es el lenguaje empleado en NodeJS. Viene impuesto como requisito para la elaboración del sistema. Se propuso el usar Golang, un lenguaje de Google, pero por temas de mantenibilidad y conocimientos del equipo se decidió que era mejor JavaScript. Uno de los puntos determinantes a la hora de elegir un lenguaje era la disponibilidad de las librerías oficiales de Stellar. Tanto Golang como JavaScript disponen de estas librerías mantenidas de forma oficial.

HTML5+CSS3

Todas las vistas de la WebApp se han elaborado mediante estos dos lenguajes, para la maquetación del sitio y para el aspecto visual del mismo. El motor de plantillas que se ha empleado con NodeJS es «ejs», que permite de una forma rápida y sencilla mostrar todo tipo de contenido sin perder la estructura HTML, ya que otros motores, al tener su propio lenguaje de marcado añaden un grado más de dificultad.

MongoDB

La necesidad de mantener los datos persistidos mediante un sistema rápido, fiable, seguro y fácil de usar nos llevó a elegir este sistema para persistir la información. MongoDB dispone de robustas librerías disponibles para NodeJS y trabajar con ellas hace que el tratamiento de datos pase a un segundo plano por la tremenda facilidad de uso. Para este proyecto se ha elegido la librería «Mongoose», que dispone de mucha documentación en Internet, lo que facilita aún más el desarrollo.

5.2. Arquitectura del sistema

Dada la complejidad del sistema y tras un período de diseño y modelado se llegó a la conclusión que lo mejor era optar por una arquitectura distribuida en varias piezas de software y no limitarse a un sistema monolítico que convertiría el proyecto en algo tedioso de mantener y modificar. Esta división se hizo teniendo en cuenta el uso del sistema y se dividió en 3 piezas fundamentales:

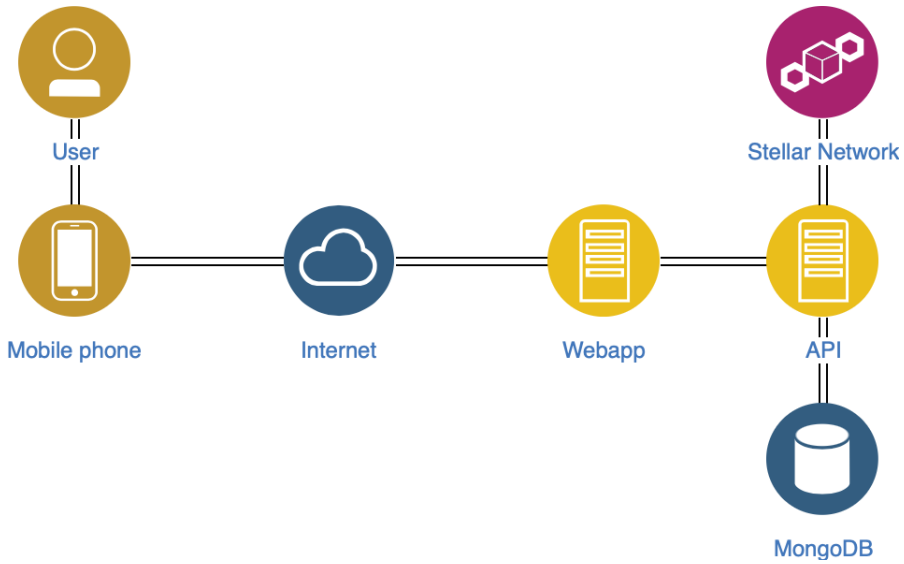

Figura 5.1: Arquitectura del sistema

- Webapp: desde la que los usuarios podrán interactuar con el sistema de forma segura y rápida.

- API: donde se desarrolla toda la lógica y se ejecutan todas las operaciones importantes. Se encarga de comunicar el resto de piezas del ecosistema y abstraer al usuario de operaciones complejas.

- Repositorio: aquí es donde se almacena toda la información tanto del propio sistema como de los usuarios.

5.2.1. Webapp

Es la pieza visible para el usuario, donde se encuentran los paneles de control desde los que realizar todas las operaciones disponibles.

En un inicio se pensó implementar esta parte como una aplicación móvil, pero por la complejidad de tener que construir una para iOS y otra para Android se optó por la forma que a nuestro parecer es la más flexible y adaptable: construir una aplicación web. La problemática actual de crear un producto para el mercado móvil es la necesidad de contar con mínimo dos aplicaciones (iOS y Android) para llegar al mayor número de usuarios posible. Haciendo una sola Webapp se garantizaba alcanzar todos los mercados pues actualmente la práctica totalidad de dispositivos móviles independientemente de la version o del sistema operativo que utilicen tienen un navegador instalado desde el que se puede acceder a la aplicación.

La estructura interna elegida está distribuida por capas. Las capas son las siguientes:

- Router: todas las peticiones HTTP llegan aquí, se procesan y se construye una respuesta que suele ser la vista solicitada por el usuario. Por la simplicidad de la lógica de esta pieza se decidió no aumentar la complejidad estructural añadiendo una capa más de lógica para gestionar las peticiones pues la mayoría son peticiones muy simples al tener toda la lógica delegada en la API.
- Vistas: es lo que el usuario ve. Se utiliza para organizar y presentar la información generada por el router de forma clara y vistosa para el usuario. Para facilitar la creación de las vistas se ha elegido un motor generador de vistas llamado EJS[1] que permite describir las plantillas para formatear la información.

Adicional mente hay que tener en cuenta la seguridad como un aspecto importante al desarrollar cualquier sistema informático, y más aún cuando este será desplegado en Internet. Por ello la Webapp no permite realizar ninguna acción importante sin estar logeado y/o validado en el sistema; de esta forma, para poder interactuar con la API es necesario loguearse en el sistema y a partir de entonces todas las peticiones serán realizadas de forma validada.

[1] `https://ejs.co/#about`

5.2.2. API

El núcleo del sistema, donde reside la lógica que realiza todas las funciones para intercomunicar todas las piezas. Recibe las peticiones mediante HTTP, las procesa y devuelve un resultado. La API pone a disposición del usuario una serie de rutas o endpoints (definidos en el apéndice) donde poder realizar las peticiones y como hacerlas. Estas peticiones pueden requerir ser validadas o no.

El código tiene una estructura por capas:

- Router: donde son recibidas todas las peticiones y desviadas al controlador adecuado, de forma transparente para el usuario. Se encarga de realizar las llamadas necesarias para chequear que las peticiones que lo requieren estén validadas correctamente, denegando el acceso en caso necesario.
- Controladores: la lógica que resuelve las peticiones recibidas desde el router. Contacta con el almacén en caso necesario para obtener o guardar información.
- Modelos: aquí se definen todas las entidades y también se encargan de realizar validaciones a la hora de interactuar con el almacén de datos. Cuando el controlador necesita guardar una entidad primero es el modelo el encargado de verificar que la información que se quiere almacenar esté correctamente formada.

Aunque estas capas a su vez pueden hacer uso de otras capas intermedias llamadas «middlewares» o «helpers». Esta forma de separar el código lo hace mucho mas mantenible y escalable para implementaciones futuras.

Los middlewares son los pasos intermedios por los que pasan algunas peticiones antes de llegar a la lógica final que formará la respuesta que espera el usuario. Un ejemplo muy claro para ayudar a entender como funciona un middleware: supongamos que se recibe una petición para eliminar un recurso del almacén, esta petición ha de ir validada y para comprobarlo hacemos uso de un middleware preparado para tal efecto, de forma que en un paso previo a la eliminación del recurso se comprobará que efectivamente quien ha realizado la petición dispone de los permisos

necesarios para hacerlo. Una característica interesante de los middlewares es que se pueden utilizar tantos como sean necesarios, de esta forma podemos hacer varias validaciones antes de llegar al propio controlador, por ejemplo: validar si el usuario esta registrado en el sistema y si dispone de los permisos apropiados para realizar un tipo de acción.

Los helpers son pequeñas porciones de código reunidas y organizadas en ficheros que permiten reutilizar cierta lógica en varios puntos del proyecto si se desea y a su vez mantener el código en general limpio y ordenado. Un ejemplo sería: durante el registro de un usuario es necesario crear un hash con su contraseña para ser almacenado en lugar del texto plano que envía el usuario; ese paso para crear el hash se externaliza en una función dentro de un fichero que trataremos como un helper.

5.2.3. Almacén de datos

Para agilizar el proceso se optó por uno de los nuevos modos de gestión de información alternativos a las bases de datos relacionales clásicas: MongoDB [2]. Ha permitido acelerar el desarrollo de las aplicaciones y centrarse en las partes realmente importantes abstrayendo casi por completo de la tarea de trabajar con la interacción de un almacén de datos. Para hacer esta interacción solo era necesario crear un modelo de datos para identificar cada entidad del sistema y mediante el uso de una librería llamada Mongoose[3], realizar todo tipo de consultas a MongoDB.

5.2.4. Stellar Network

Se denomina Stellar Network al conjunto de funcionalidades que hacen posible el funcionamiento de la criptomononeda Lumens desarrollada por la organización Stellar. Esta tecnología es de código abierto, distribuida y de propiedad de la comunidad.

La estructura es simple, se divide en dos piezas. Por un lado cuentan con una API llamada

[2] `https://www.mongodb.com/what-is-mongodb`
[3] `https://mongoosejs.com/`

Horizon[4] que permite enviar transacciones a la red y verificar el estado de las cuentas de forma sencilla. Por otro lado tienen el llamado Stellar Core[5], un software ejecutado por diversos servidores. Estos servidores son mantenidos por diferentes individuos y entidades. Su función principal es mantener una copia local del libro mayor, comunicarse y estar sincronizado con otras instancias de Stellar Core en la red.

Stellar cuenta con dos redes, una publica donde se hacen transferencias reales entre individuos y otra red denominada «horizon-testnet» que como su nombre indica es una red para hacer tests y que los Lumens que circulan en ella no tienen valor alguno. Esta red ha sido muy útil antes de empezar a hacer llamadas reales desde la aplicación. Permitió hacer pruebas de cualquier tipo sin ningún riesgo gracias a Stellar Laboratory, una herramienta web desarrollada por Stellar. En ella fue posible explorar los endpoints de la API, hacer transacciones, firmarlas y ver que se hacían efectivas en la o las respectivas cuentas.

Todo el desarrollo para este Trabajo de Fin de Grado se ha hecho contra la «horizon-testnet» debido a que permite conseguir un funcionamiento idéntico al que conseguiríamos en la red publica de Stellar pero sin costes ni riesgos. Si en algún momento se quisiera cambiar de la red de pruebas a la red publica tan solo habría que modificar una sola variable en toda la aplicación.

Uso de Stellar Network en la aplicación

Una de las partes más importantes del presente trabajo ha sido la de implementar un sistema de pago de recompensas haciendo uso de la red de Stellar. Gracias a Stellar hemos podido implantar un sistema de pagos seguros, rápidos y potentes.

Cada vez que un usuario se registra en la plataforma se crea una cuenta para el, que consiste en un par de claves; una secreta que servirá para firmar transacciones y otra pública que será la empleada para recibir pagos. Este sistema permite realizar un pago con solo conocer la clave pública de la cuenta a la que queremos enviar el pago. La analogía para los pagos dentro de nuestra aplicación es el concepto de «validación de recompensas» que será descrito más adelante.

[4] `https://www.stellar.org/developers/horizon/`
[5] `https://www.stellar.org/developers/stellar-core`

Cuando un turista consigue validar todos los puntos que conforman una de las rutas propuestas, tiene la posibilidad de validar su recompensa: de esta forma el sistema verifica que efectivamente el turista ha pasado por todos los puntos y realiza el pago desde la cuenta de la institución que ha creado la recompensa.

De esta forma se ha conseguido un sistema autónomo donde varios usuarios pueden interactuar con la seguridad de que los pagos se realizan de forma correcta y eficaz.

A continuación, debido a la importancia de Stellar en este trabajo se incluyen los fragmentos de código críticos para su utilización:

Para hacer uso de la infraestructura de Stellar se hace uso de la librería proporcionada por la comunidad de Stellar, pública y gratuita, para ello basta con importarla a nuestro código:

```
1 const StellarSdk = require('stellar-sdk');
2 const server = new StellarSdk.Server('https://horizon-testnet.stellar.org');
```

Cuando una nueva cuenta de usuario es creada, también interesa que sea generada una cuenta en Stellar con la que poder realizar las futuras operaciones. Con la siguiente función se crea el par de claves necesario para poder realizar operaciones dentro de la red de Stellar. Esta funcion es llamada siempre que una nueva cuenta de usuario es dada de alta en el sistema:

```
1  exports.createStellarAccount = async function (account) {
2      const pair = StellarSdk.Keypair.random();
3
4      try {
5          const response = await fetch(
6              `https://friendbot.stellar.org?addr=${encodeURIComponent(pair.
                   publicKey())}`
7          );
8          await response.json();
9
10         account.stellarPublic = pair.publicKey();
11         account.stellarSecret = pair.secret();
12     } catch (e) {
13         return res.status(500).send('Error creating account: ' + e);
14     }
```

```
};
```

Para mostrar toda la información relativa a una cuenta de Stellar lo hacemos con la siguiente función:

```
exports.getAccount = async function (req, res) {
    try {
        const account = await server.loadAccount(req.params.accountid);
        return res.status(200).send(account);
    } catch (e) {
        return res.status(500).send('Error getting account: ' + e);
    }
};
```

La función más importante y que contiene la lógica para la emisión de los pagos. Es importante remarcar la facilidad que otorga el uso de dicha librería y que nos permite de una forma limpia, segura y rápida implementar soluciones de pago como esta.

```
exports.sendPayment = async function (sourceAccountID, destinationAccountID,
    amount) {
    let sourceAccount = await Account.findById(sourceAccountID);
    if (!sourceAccount) return new Error("Source account not found.");

    const sourceKeyPair = StellarSdk.Keypair.fromSecret(sourceAccount.
        stellarSecret);
    const sourcePublicKey = sourceKeyPair.publicKey();

    let destinationAccount = await Account.findById(destinationAccountID);
    if (!destinationAccount) return new Error("Source account not found.");

    const destinationKeyPair = StellarSdk.Keypair.fromSecret(destinationAccount
        .stellarSecret);
    const destinationPublicKey = destinationKeyPair.publicKey();

    StellarSdk.Network.useTestNetwork();

    const account = await server.loadAccount(sourcePublicKey);
    const fee = await server.fetchBaseFee();
```

```
18
19      let aux = amount + ".0000000";
20
21      const transaction = new StellarSdk.TransactionBuilder(account, { fee })
22          .addOperation(StellarSdk.Operation.payment({
23              destination: destinationPublicKey,
24              asset: StellarSdk.Asset.native(),
25              amount: aux,
26          }))
27          .setTimeout(30)
28          .build();
29
30      transaction.sign(sourceKeyPair);
31
32      try {
33          await server.submitTransaction(transaction);
34          return true;
35      } catch (e) {
36          return e;
37      }
38  };
```

Cuando un usuario ha completado todos los puntos de prueba de paso exigidos en una ruta, puede solicitar la validación de la recompensa para obtener la cantidad establecida:

```
1   exports.validate = async function (req, res) {
2       try {
3           let reward = await Reward.findOne({
4               name: req.body.name
5           });
6           if (!reward) return res.status(404).send({
7               error: 'reward not exists'
8           });
9
10          reward.beacons.forEach(function (beacon) {
11              if (!beacon.validado) {
12                  return res.status(400).send({
13                      error: 'all beacons must be validated',
```

```
14                      message: 'all beacons must be validated before validate a
                            reward'
15                  });
16              }
17          });
18
19          reward.terminada = true;
20          await reward.save();
21
22          sendPayment(reward.account_id, req.body.account_id, reward.recompensa);
23          return res.status(200).send(true);
24      } catch (e) {
25          return res.status(500).send({
26              error: 'error validating reward',
27              message: e.message
28          });
29      }
30  };
```

Y el último paso, que es cuando el turista se acerca al punto designado para la obtención de la recompensa y la persona encargada realiza la accion:

```
1   exports.claim = async function (req, res) {
2       try {
3           let reward = await Reward.findOne({
4               name: req.params.name
5           });
6           if (!reward) return res.status(404).send({
7               error: 'reward not exists'
8           });
9
10          let result = sendPayment(reward.account_id, req.body.hostelero_id,
                reward.recompensa);
11          if (!result) return res.status(400).send({
12              error: 'error sending payment for claim reward',
13              message: 'error sending payment for claim reward'
14          });
15
```

```
16          reward.canjeada = true;
17          reward.hostelero_id = req.body.hostelero_id;
18          await reward.save();
19
20          return res.status(200).send({
21              message: 'reward claimed successfully'
22          });
23      } catch (e) {
24          return res.status(500).send({
25              error: 'error claiming reward',
26              message: e.message
27          });
28      }
29  };
```

Capítulo 6

Casos de uso

En esta sección se detallan las acciones que permite el sistema según el actor. Algunas de estas acciones son generales por lo que se han incluido todas en una sección a fin de no repetir para cada actor. Además, por cada caso de uso se referencian los requisitos funcionales que se han incluido como apéndice en el Capítulo 8.

6.1. Actores del sistema

Los actores contemplados son:

El **turista**, que es el usuario final de la aplicación, es decir quien disfruta de todas sus posibilidades. Buscar rutas, subscribirse a ellas para mas tarde poder validar los puntos de prueba de paso, buscar puntos donde validar su recompensa y por ultimo validarla.

El **hostelero**, es un usuario intermedio, esto quiere decir que se beneficia de la aplicación pero a su vez hace posible que el usuario final pueda disfrutarla. Este actor podrá crear puntos donde validar recompensas y validar las recompensas de los turistas que lo soliciten.

La **institución**, es un usuario de control y administración de la aplicación, hace posible y da el servicio al hostelero y el turista. Tendrá total control en cuanto a crear, modificar y eliminar rutas, puntos de prueba de paso y puntos de recompensa.

6.2. Generales

6.2.1. Crear cuenta

Todo usuario que quiera interactuar con el sistema necesita una cuenta de usuario. En este proceso se introducen los datos requerido y se selecciona el tipo de usuario deseado. En este paso también se genera una cuenta en la red de Stellar y se almacena con los datos de la cuenta. Ver tabla 8.1

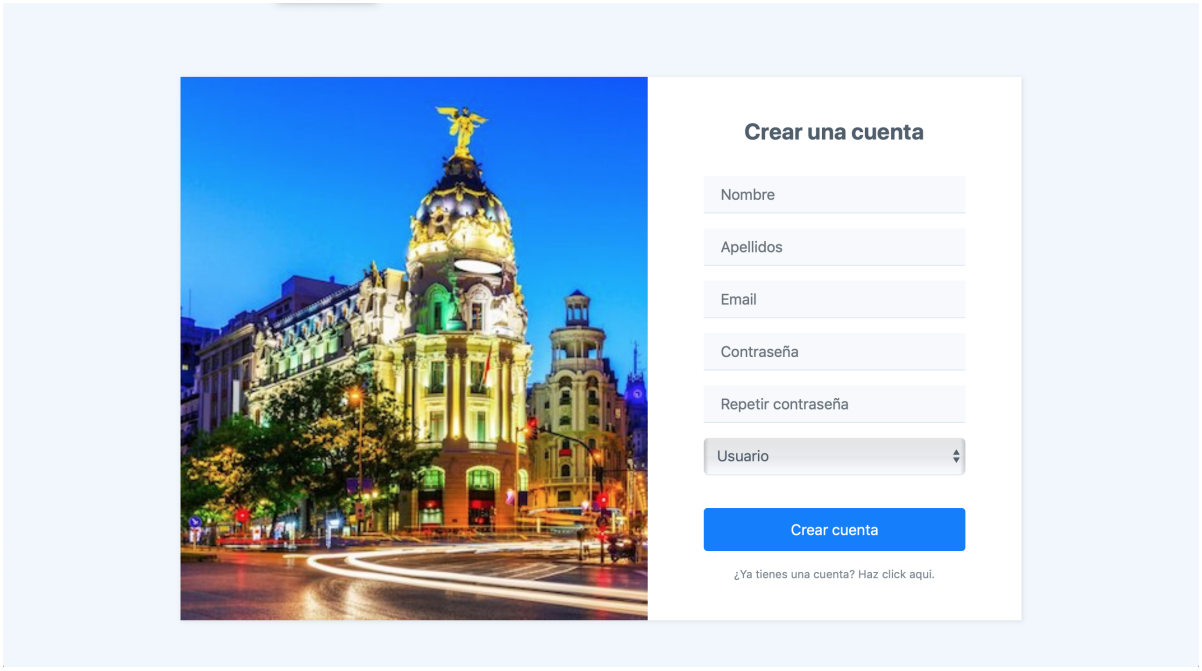

Figura 6.1: Creación de una cuenta

6.2.2. Acceso al sistema

Todo usuario que quiera interactuar con el sistema tiene que haber introducido sus credenciales antes. Mediante el proceso de login, el usuario introduce su nombre y contraseña y es validad por el sistema. Ver tabla 8.2

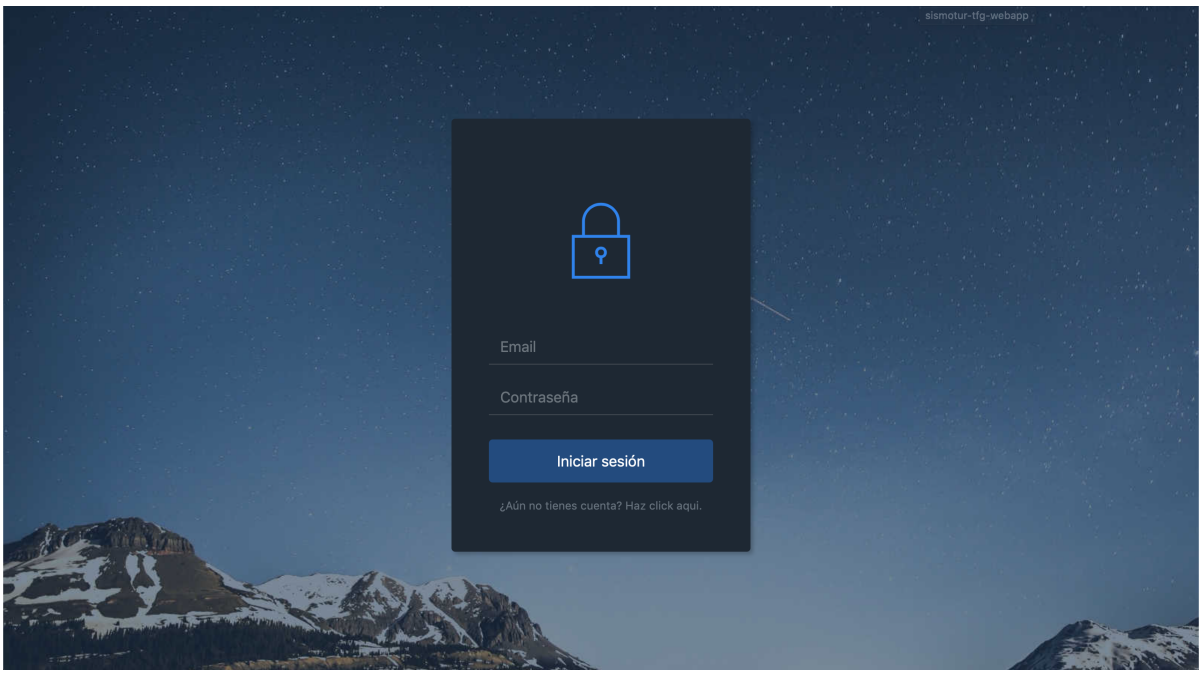

Figura 6.2: Acceso al sistema

6.2.3. Cerrar sesión

Cuando un usuario quiere dejar de usar la aplicación, puede cerrar la sesión actualmente abierta. Ver tabla 8.3

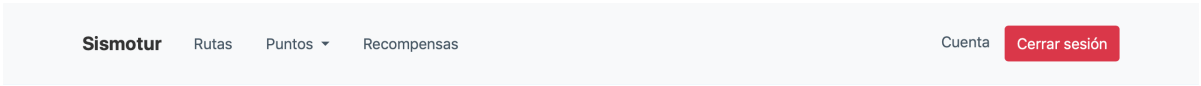

Figura 6.3: Cerrar la sesión

6.3. Turista

6.3.1. Suscribirse a una ruta

Cuando un turista llega a su destino turístico y se plantea realizar una de las rutas incluidas en el sistema lo primero que tiene que hacer es suscribirse a una de estas rutas, puede hacerlo desde el listado de rutas. Ver tabla 8.5

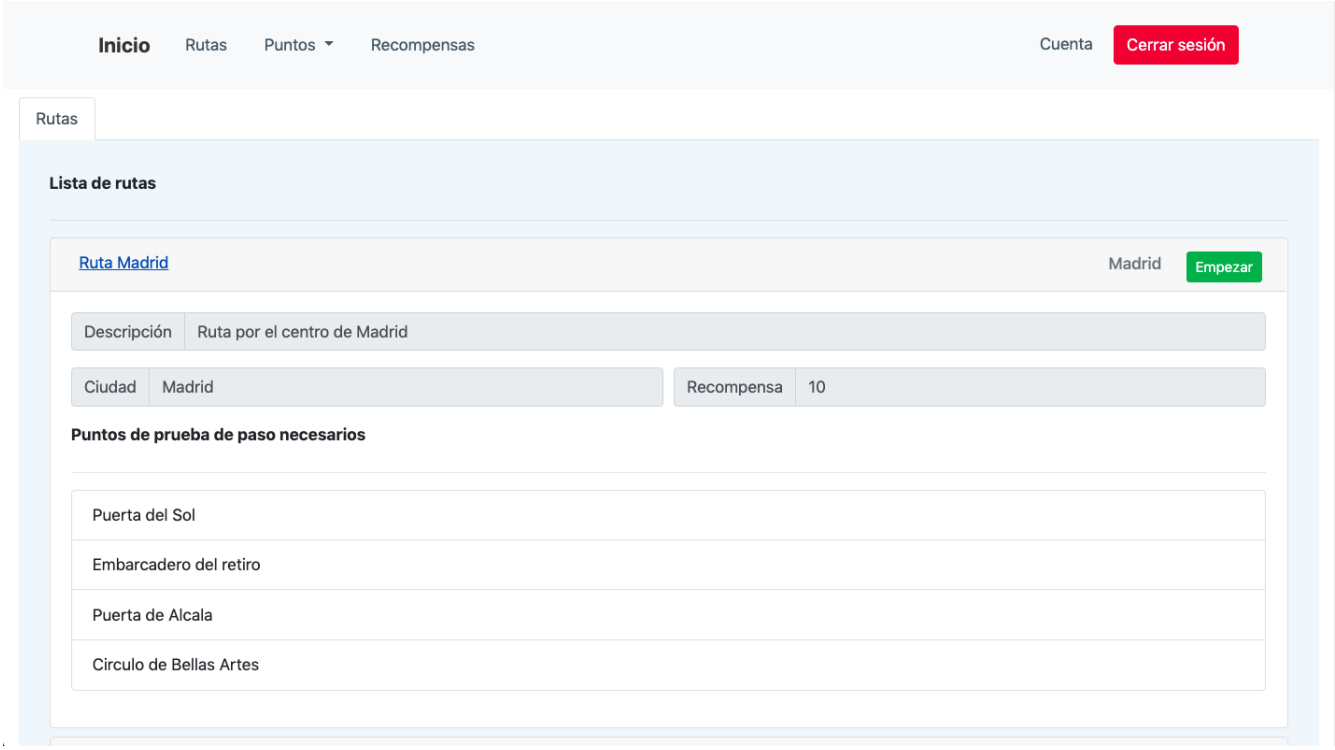

Figura 6.4: Suscribirse a una ruta

6.3.2. Validar punto de prueba de paso

A medida que va pasando por cada uno de los puntos que conforman la ruta, el turista puede ir validando mediante la aplicación su progreso. Puede seleccionar cada uno de estos puntos desde el listado de puntos de prueba de paso. Ver tabla 8.6

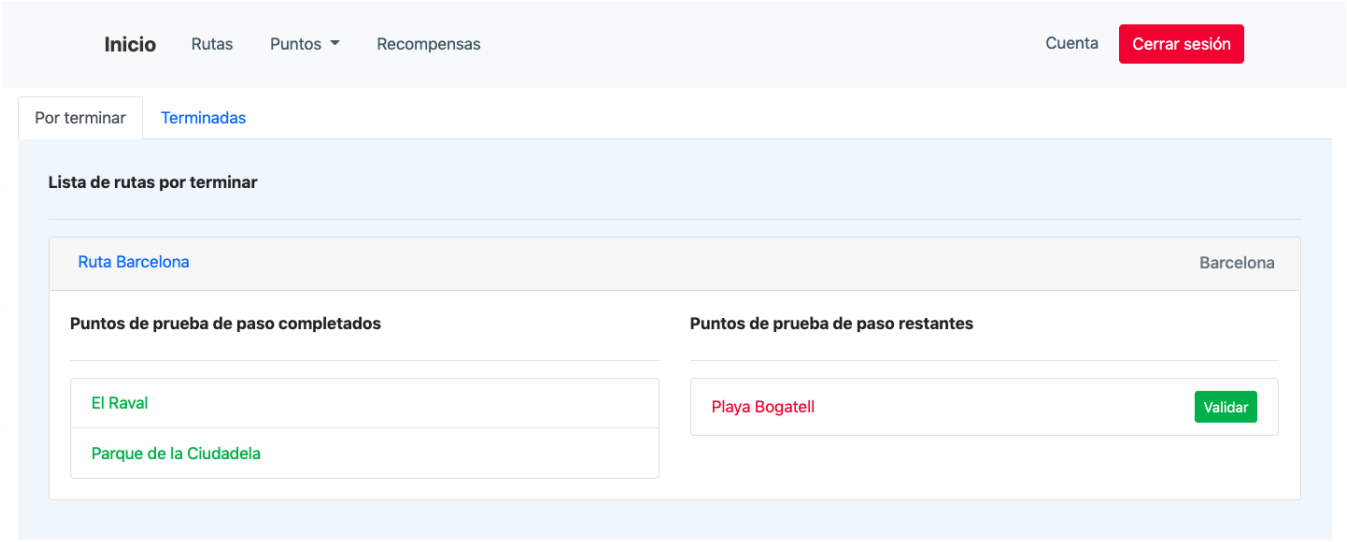

Figura 6.5: Validar punto de prueba de paso

6.3.3. Validar recompensa

Una vez que ha pasado por todos los puntos de prueba de paso y estos han sido correctamente validados, el turista querrá obtener su recomensa, para lo cual debe acudir a uno de los puntos autorizados disponibles y validad su recompensa con el hostelero, quien hará las comprobaciones oportunas. Cuando se realiza la validación, el sistema llama a la red de Stellar para procedere a la realización del pago a la cuenta de Stellar asociada con el usuario. Ver tabla 8.10

Durante el proceso de validación de una recompensa se realizan múltiples llamadas a la red de Stellar, la primera es realizada cuando el usuario realiza la validacion de la recompensa, el sistema comprueba que todos los puntos de la ruta (beacons) han sido alcanzados y entonces realiza el pago de la cantidad designada a la cuenta Stellar del usuario. Cuando el usuario acude al punto definido para recoger la recompensa el hostelero hara la operación inversa y nuevamente el sistema hará la llamada a la red de Stellar para realizar una nueva transacción, pero esta vez desde la cuenta del turista a la del hostelero. En este punto el hostelero hará entrega de la recompensa definida. A continuación se detalla un ejemplo de la comunicación con Stellar:

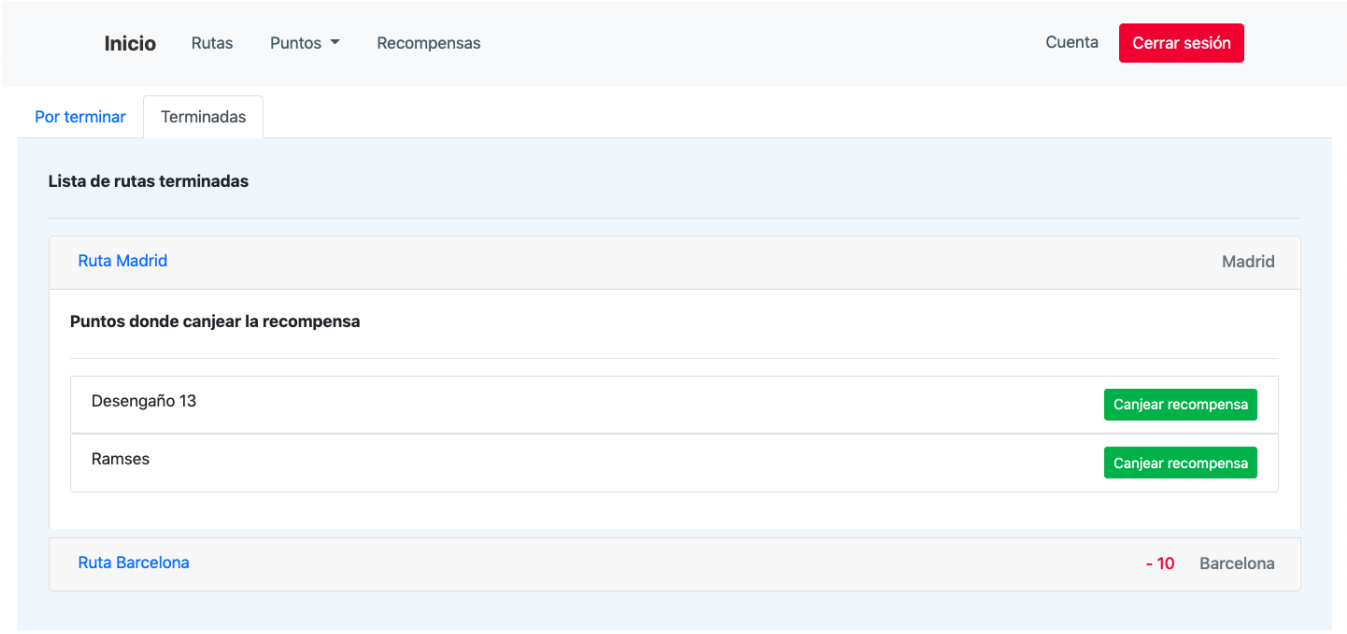

Figura 6.6: Validar recompensa

Supongamos las siguientes cuentas:

```
{
  "PUBLIC KEY A": "GB2B4H4DLOM3JK6UQ2ZRKGX67JO2C5QXUPC2HHUS54D4XY4YG235YJUK",
  "SECRET KEY A": "SCOBUGNCZTVQXBTAMV7NI7VOW43YKIORK5L2GV3GMEHA7S52TVTN3XOK",
  "PUBLIC KEY B": "GBTNPAQHHHBBAGO4Q5P7CIEKHIZHCSO6DNH3IARNY2TJXPAP4KUGB3WN",
  "SECRET KEY B": "SDFTQXKAV5C7ULBCSVKOZPH7V2Z5VLD7U7AAFQ322LBE57BR3W4U4R4G"
}
```

Si consultamos la informacion de la primera cuenta obtenemos la siguiente información:

```
{
  "id": "GB2B4H4DLOM3JK6UQ2ZRKGX67JO2C5QXUPC2HHUS54D4XY4YG235YJUK",
  "account_id": "GB2B4H4DLOM3JK6UQ2ZRKGX67JO2C5QXUPC2HHUS54D4XY4YG235YJUK",
  "sequence": "2806271501664256",
  "subentry_count": 0,
  "last_modified_ledger": 653386,
  "thresholds": {
    "low_threshold": 0,
    "med_threshold": 0,
```

```
    "high_threshold": 0
  },
  "flags": {
    "auth_required": false,
    "auth_revocable": false,
    "auth_immutable": false
  },
  "balances": [
    {
      "balance": "10000.0000000",
      "buying_liabilities": "0.0000000",
      "selling_liabilities": "0.0000000",
      "asset_type": "native"
    }
  ],
  "signers": [
    {
      "weight": 1,
      "key": "GB2B4H4DLOM3JK6UQ2ZRKGX67JO2C5QXUPC2HHUS54D4XY4YG235YJUK",
      "type": "ed25519_public_key"
    }
  ],
  "data": {}
}
```


Ahora vamos a realiza una transacción para ver la respuesta que obtenemos de la red de Stellar. Primero se firma la transaccion que se quiere realizar:


```
{
  "sign": "AAAAAHQeH4NbmbSr1IazFRr++l2hdhejxaOeku8Hy+OYNrfcAAAAZAAJ",
  "hash": "1cc5f9141cfa119b874bda8169704fca9a862422e9b8a2d3647468933f3d846d"
}
```


Y posteriormente ya se puede incluir dicha transacción a la red de Stellar.


```
{
  "_links": {
    "transaction": {
      "href": "https://horizon-testnet.stellar.org/transactions/1
```

```
            cc5f9141cfa119b874bda8169704fca9a862422e9b8a2d3647468933f3d846d"
      }
    },
    "hash": "1cc5f9141cfa119b874bda8169704fca9a862422e9b8a2d3647468933f3d846d",
    "ledger": 653548,
    "envelope_xdr": "AAAAAHQeH4NbmbSr1IazFRr++l2hdhejxaOeku8Hy+OYNrfcAAAAZAAJ",
    "result_xdr": "AAAAAAAAAGQAAAAAAAAAAQAAAAAAAAABAAAAAAAAAAA=",
    "result_meta_xdr": "AAAAAQAAAAIAAAADAAn47AAAAAAAAAAAdB4fg1uZtKvUhrMVGv76Xa"
}
```

6.4. Hostelero

6.4.1. Crear nuevo punto de recompensa

El hostelero es el encargado de dar de alta su negocio como punto de recompensa, que es donde el usuario acudirá una vez ha terminado de recorrer una ruta. Ver tabla 8.13

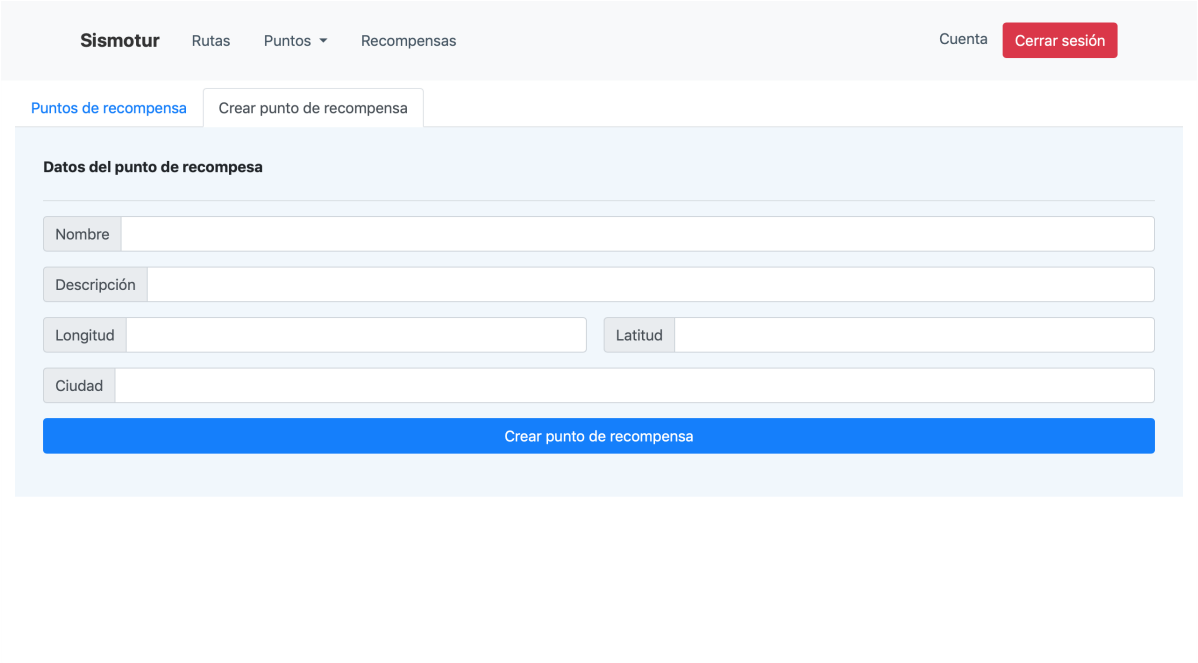

Figura 6.7: Crear punto de recompensa

6.4.2. Obtener listado de puntos de recompensa

El hostelero puede obtener un listado de los puntos de recompensa creados por el. Ver tabla 8.14

6.4.3. Borrar punto de recompensa

Cuando se deja de usar un punto de recompensa, puede ser eliminado del sistema. Ver tabla 8.16

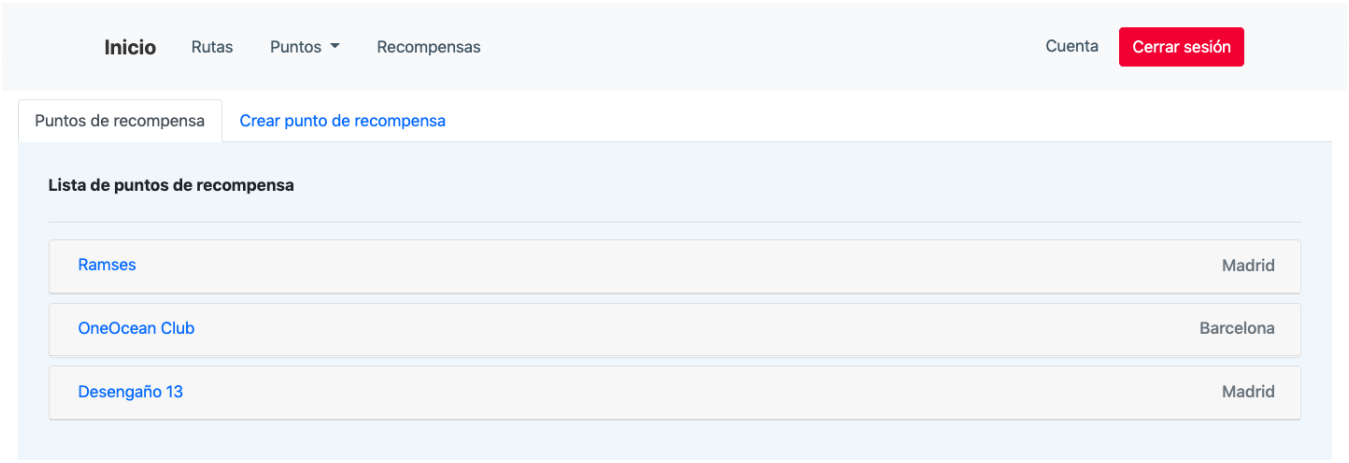

Figura 6.8: Listado puntos de recompensa

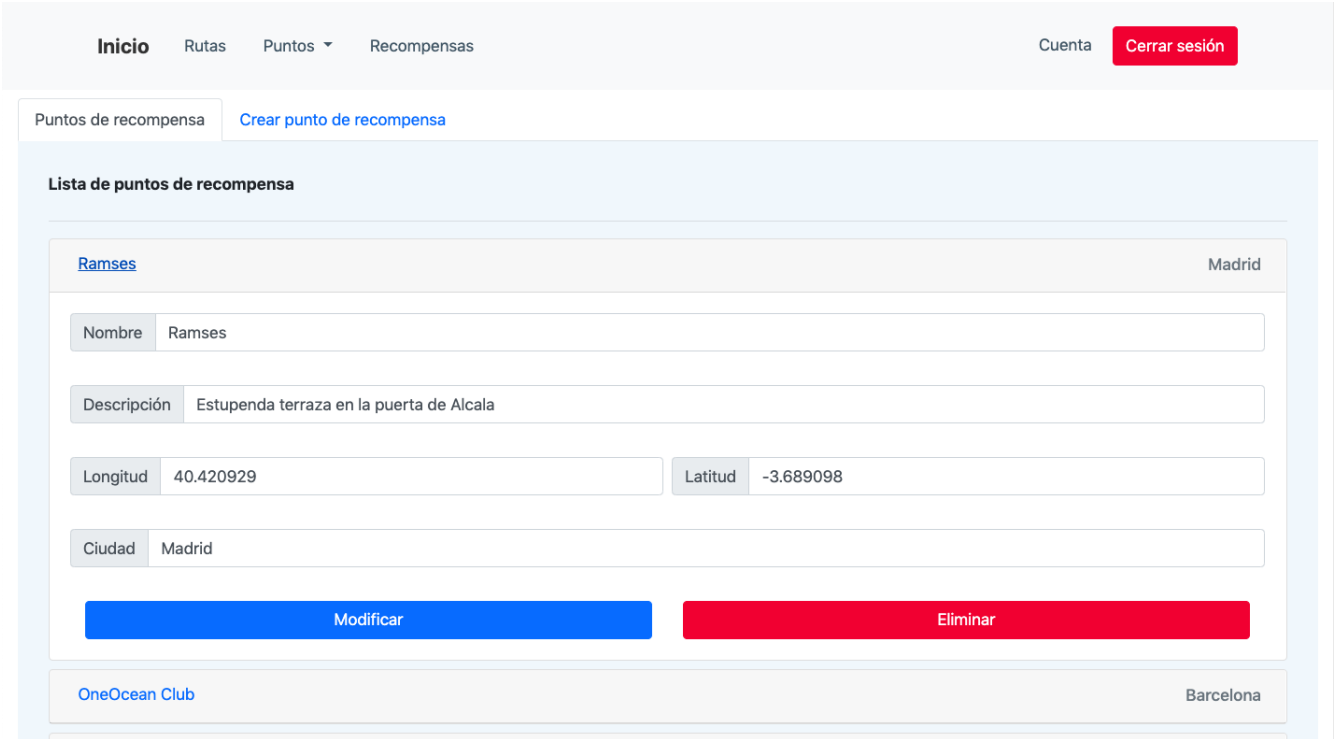

Figura 6.9: Borrar punto de recompensa

6.5. Institución

6.5.1. Crear un nuevo punto de prueba de paso

Para crear los puntos de prueba de paso que forman una ruta. Ver tabla 8.22

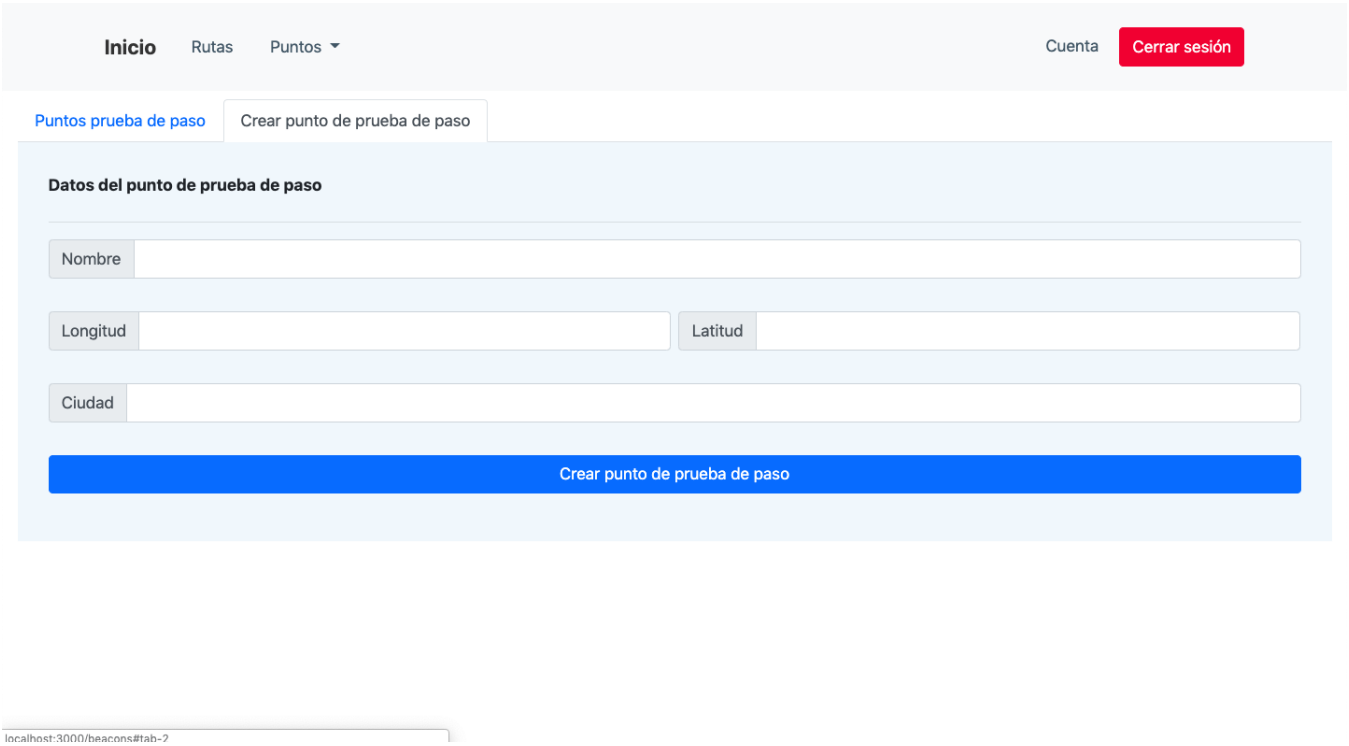

Figura 6.10: Crear un punto de prueba de paso

6.5.2. Obtener listado de puntos de prueba de paso

Se puede obtener un listado con todos los puntos de prueba de paso que forman una ruta. Ver tabla 8.23

6.5.3. Borrar punto de prueba de paso

Si algún punto deja de ser necesario por cualquier motivo, este puede ser eliminado del sistema. Ver tabla 8.25

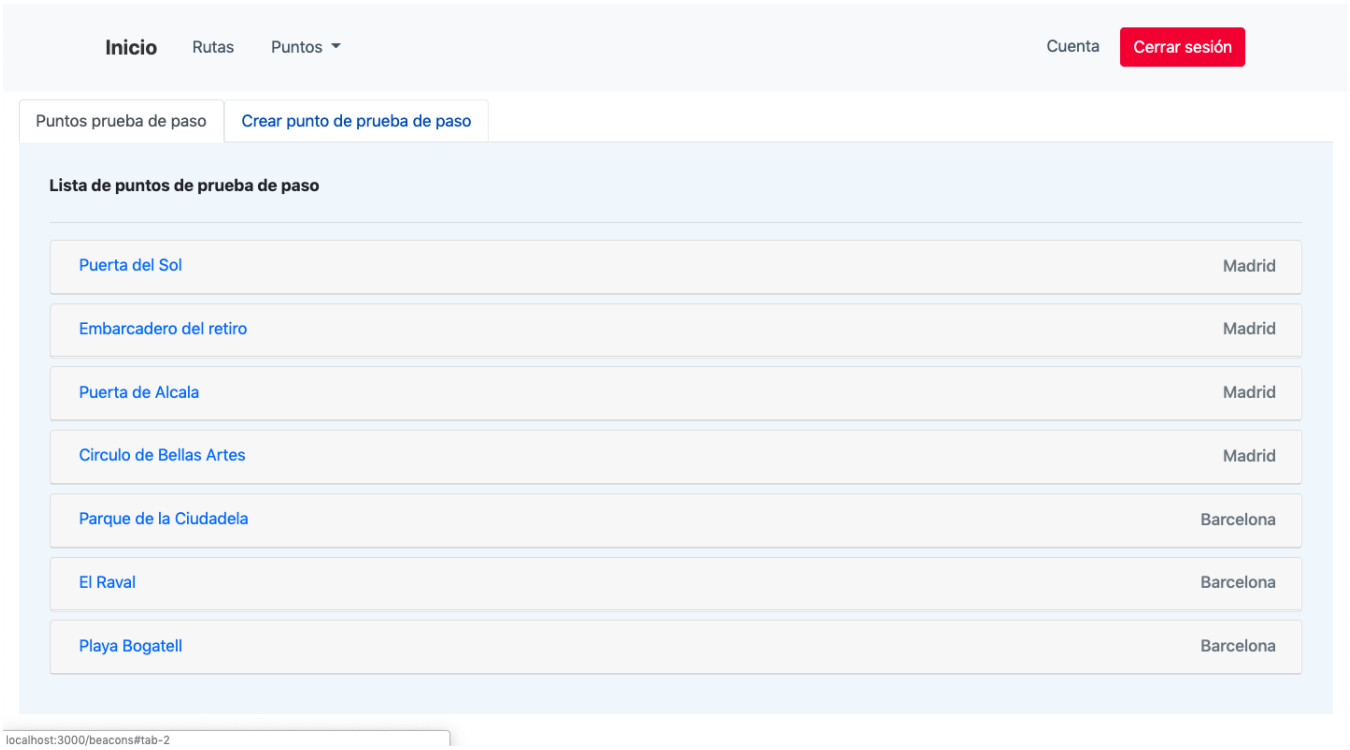

Figura 6.11: Obtener listado de puntos de prueba de paso

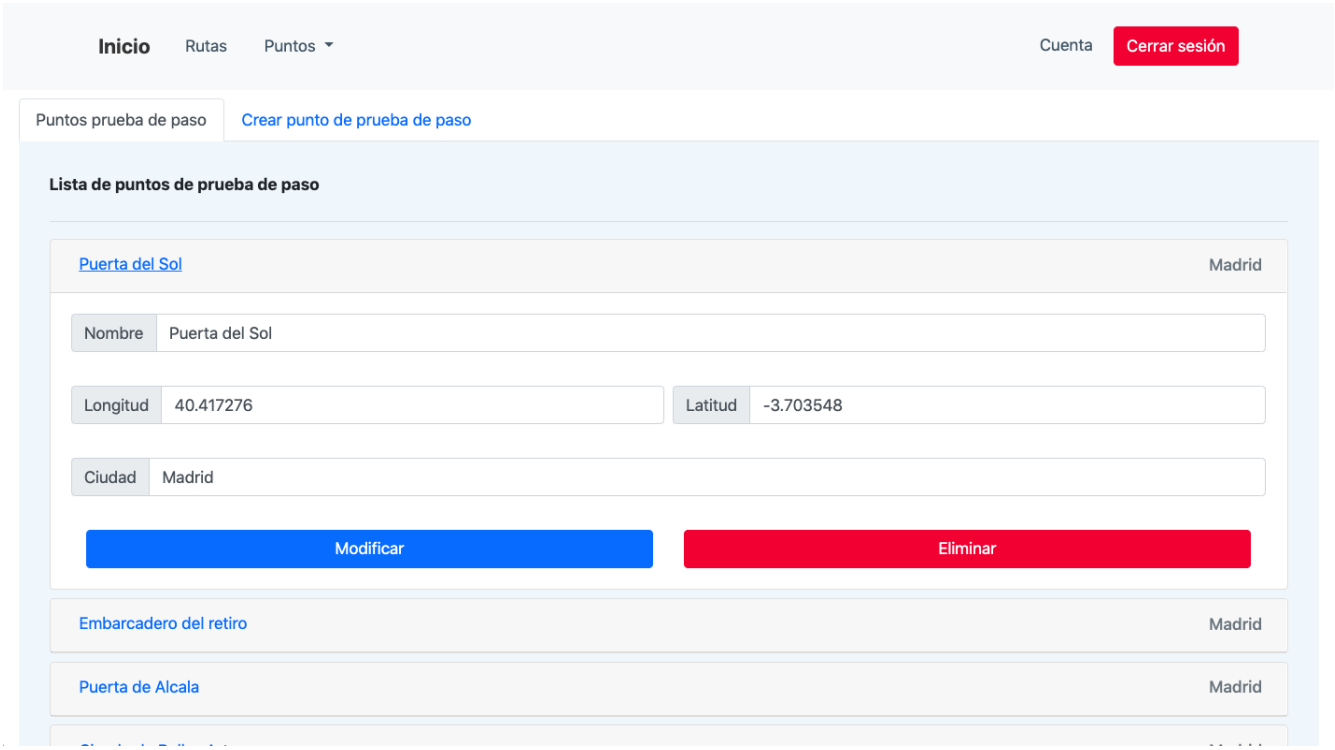

Figura 6.12: Borrar punto de prueba de paso

Capítulo 7

Conclusiones

7.1. Resumen

Este trabajo ha permitido profundizar y ampliar las materias estudiadas a lo largo del grado y el estudio de otras, que por su novedad no están incluidas en el plan de estudios del Grado en Ingeniería del Software. La oportunidad de tomar decisiones a la hora de diseñar el sistema, en ocasiones han llevado a situaciones problemáticas, generalmente causadas por desconocimiento de la materia, lo cual ha conseguido generar cierta creatividad a la hora de afrontar problemas planteando soluciones a estos que no se conocían, es decir, se ha desarrollado la capacidad de ser autosuficiente y diseñar soluciones a problemas que no habían sido planteados en el grado. Esta es una habilidad que sólo se consigue con años de experiencia.

A lo largo de la elaboración del proyecto se ha pasado por todas las etapas de creación de un producto software, lo que ha permitido también adquirir competencias técnicas en este aspecto, sumando la capacidad necesaria para no solo ser capaz de escribir un código que funcione, si no de diseñar, planificar e implementar toda una solución software para una necesidad real y concreta.

Muchas de las tecnologías empleadas en el proyecto ya eran conocidas previamente gracias a muchas de las materias estudiadas, posibilitando enormemente la elaboración de un producto

más sólido y escalable al no tener que estudiar de cero esos apartados.

Haber podido completar todos los objetivos propuestos al inicio del proceso ha supuesto una gran satisfacción. Esto ha permitido reforzar ciertas carencias en algunos apartados como el diseño y la planificación, que aunque hay varias asignaturas en el grado destinadas a ello, pueden no ser suficientes para un ingeniero de software.

7.2. Competencias adquiridas

El desarrollo de este Trabajo de Fin de Grado me ha permitido afianzar y adquirir una serie de nuevos conocimientos, no solo tecnológicos si no en relación con el mundo laboral.

- JavaScript fue uno de los requisitos para implementar el sistema, no fue problema ya que habíamos trabajado antes con este lenguaje en varias asignaturas. Con esta base clara tuve que elegir una arquitectura que fuera modular y escalable, por eso decidí separar la lógica en una API. Dado que era la primera vez que desarrollaba este tipo de arquitectura podemos decir que me ha permitido adquirir nuevas capacidades.
- La comunicación con Stellar Network se ha hecho a través de su API Horizon. Nunca había trabajado con APIs de terceros, pero la documentación para desarrolladores que ofrece Stellar me ha facilitado mucho esta tarea. Con esto también me he dado cuenta de la importancia que tiene el desarrollo de una buena documentación cuando desarrollas una API que quieres que sea usada por otras personas. Aunque finalmente no he usado Ethereum como criptomoneda para el proyecto si que he investigado sobre la forma de implementarlo y he de decir que la documentación sobre su API no era tan extensa y detallada como la de Stellar.
- Durante la carrera hemos desarrollado proyectos con una base solida en cuanto a requisitos y funcionalidades. En cambio al enfrentarme a un proyecto real y que en un principio no estaba totalmente definido me he dado cuenta de la importancia que tiene hacer una buena ingeniería previa al desarrollo del producto, definir los requisitos y funcionalidades para evitar que en una etapa mas avanzada del proyecto surjan cambios que retrasen el desarrollo.

7.3. Trabajo futuro

Es este apartado describimos una serie de ideas posibles para mejorar la aplicación.

En primer lugar, esta prueba de concepto se debería integrar en la aplicación Inventrip de Sismotur, tanto en su versión web como en iOS y en Android. Puesto que es para lo que esta pensada. Una vez integrada las posibilidades crecen de manera notable.

- Implementar un sistema de mapas y GPS. Google Maps Platform ofrece las APIs y los SDKs necesarios para poder llevar a cabo las siguientes funciones:
  - Ver las rutas sobre el mapa y que la aplicación nos pueda guiar desde donde estemos hasta el inicio de la ruta y de ahí hasta el final.
  - Ver todos los puntos de prueba de paso sobre el mapa, por si queremos explorar por nuestra cuenta y visitar alguno de ellos. La aplicación también nos guiaría hasta el punto elegido.
  - Ver todos los puntos de recompensa sobre el mapa y que la aplicación nos pueda guiar desde donde estemos hasta el bar, restaurante, mercado...
- Implementar un sistema de valoración para que los usuarios puedan dar su opinión acerca de las rutas realizadas para incentivar el uso de la aplicación.
- Dar visibilidad a las subvenciones que financian las recompensas de las rutas.
- Implementar el sistema para que el administrador pueda dar de alta y controlar quien tiene perfil de hostelero y quien no, dado que ahora no se tiene control sobre el como se registran los usuarios, ya sean administrador, hostelero o usuario.

7.4. Summary

It has been very interesting for us to propose a solution to a real problem, go a little beyond what we are used to in the subjects of the career.

Regarding the technologies used for the development of the application we have not had any major problems, either we had already worked with them during the career or one of the two knew a little more and we taught each other. The biggest challenge we would say has been in terms of cryptocurrencies. Although we were already a little informed before starting the project on this subject, the knowledge we had was not enough to carry out the project. We study and compare both in characteristics and at low level various types of cryptocurrencies until we reach Stellar. It is undoubtedly the one that best suited the project.

7.5. Knowledge acquired

The development of this Final Degree Project has allowed us to consolidate and acquire a series of new knowledge, not only technological but in relation to the world of work.

- JavaScript was one of the requirements to implement the system, for us it was not a problem since we had worked with this language before. With this base we had to choose an architecture that was modular and scalable, that's why we decided to separate the logic in an API. Since it was the first time we developed this type of architecture we can say that it has allowed us to acquire new capabilities.
- Communication with Stellar Network has been done through its Horizon API. We had never worked with third-party APIs, but the developer documentation offered by Stellar has made this task much easier. With this we have also realized the importance of developing good documentation when you develop an API that you want to be used by other people. Although finally we don't use Ethereum as cryptocurrency for the project if we investigate on how to implement it and we have to say that the documentation on its API was not as extensive and detailed as Stellar's.
- Regarding the work part, we can say that our feet have been sitting on the ground a bit. During the career we have developed projects with a solid base in terms of requirements and functionalities. On the other hand, when facing a real project and that at first was not totally defined, we have realized the importance of doing a good engineering prior to the development of the product, defining the requirements and functionalities to avoid changes in a more advanced stage of the project that delay development.

7.6. Future work

In this section we will describe a series of possible ideas to improve the application.

First, this proof of concept should be integrated into the Inventrip application of Sismotur, in its web version, in iOS and Android. It was thought for that. Once integrated, the possibilities grow significantly.

- Implement a map and GPS system. Google Maps Platform offers the necessary APIs and SDKs to carry out the following functions:
  - See the routes on the map and the application can guide us from where we are until the beginning of the route and from there to the end.
  - See all the proof of pass points on the map, in case we want to explore on our own and visit any of them. The application would also guide us to the chosen point.
  - See all the reward points on the map and the application can guide us from where we are to the bar, restaurant, market ...
- Implement a valuation system so that users can comment on the routes carried out to encourage the use of the application.
- Give visibility to the grants that finance the rewards of the routes.
- Implement the system so that the administrator can register and control who has a hotelier profile and who does not, given that now there is no control over how users register, whether they are an administrator, a hotelier or a user.

Capítulo 8

Apéndices

8.1. Requisitos funcionales

A continuación se detallan los requisitos funcionales de cada actor para tener claro que puede hacer cada uno dentro de la aplicación. Teniendo claro esto sera mas fácil entender la funcionalidad de la aplicación completa.

8.1.1. Generales

A continuación se detallan los requisitos generales, que aplican de igual forma para todos los usuarios.

Cuadro 8.1: Crear cuenta

Crear cuenta

Descripción	El usuario quiere crear una cuenta para poder interactuar con el sistema.
Precondición	–
Secuencia	1. El usuario accede a la pagina de creación de cuenta. 2. El usuario introduce todos los datos del formulario. 3. El sistema verifica que todos los datos son válidos. 4. El sistema crea la nueva cuenta y la almacena.
Errores	1. Si los datos introducidos por el usuario son incorrectos, la cuenta no se crea y se informa al usuario.
Postcondición	La cuenta es creada y almacenada.
Notas	Existen tres tipos de cuenta de usuario: ▪ Turista ▪ Hostelero ▪ Institución

Cuadro 8.2: Acceder al sistema

Acceder al sistema

Descripción	El usuario quiere acceder al sistema con sus credenciales.
Precondición	El usuario está registrado en el sistema.
Secuencia	1. El usuario accede a la página de login. 2. El usuario introduce sus datos de acceso. 3. El sistema verifica que el usuario existe y los datos son correctos y realiza el proceso de login.
Errores	1. Si los datos introducidos por el usuario no son correctos el sistema devuelve a la pagina de login.
Postcondición	Se genera un nuevo token que es enviado al usuario para que pueda hacer peticiones de forma validada.
Notas	

Cuadro 8.3: Cerrar sesión

Cerrar sesión

Descripción	El usuario quiere terminar la sesión.
Precondición	El usuario está logeado en el sistema.
Secuencia	1. El usuario pulsa el botón para terminar la sesión. 2. El sistema elimina la sesión del usuario.
Errores	–
Postcondición	La sesión es eliminada del sistema.
Notas	

8.1.2. Turista

Cuadro 8.4: Consultar rutas disponibles

Consultar rutas disponibles

Descripción	El usuario quiere conocer las rutas disponibles.
Precondición	El usuario está logeado en el sistema.
Secuencia	1. El usuario accede a la vista de rutas disponibles. 2. El sistema realiza una búsqueda y muestra las rutas disponibles.
Errores	–
Postcondición	–
Notas	

Cuadro 8.5: Suscribirse a una ruta

Suscribirse a una ruta

Descripción	El usuario quiere suscribirse a una ruta.
Precondición	El usuario está logeado en el sistema.
Secuencia	1. El usuario accede a la vista de rutas disponibles. 2. El usuario selecciona una ruta disponible para suscribirse. 3. El sistema registra la ruta que ha elegido el usuario.
Errores	–
Postcondición	La ruta elegida por el usuario queda relacionada con este.
Notas	

Cuadro 8.6: Validar punto de prueba de paso

Validar punto de prueba de paso

Descripción	El usuario quiere validar un punto de prueba de paso al que ha llegado.
Precondición	El usuario está logeado en el sistema y tiene el código de validación del punto de prueba de paso.
Secuencia	1. El usuario accede a la vista de las rutas a las que está suscrito. 2. El usuario selecciona el punto de prueba de paso y pulsa el botón para validar. 3. El sistema verifica que los datos sean correctos y registra la validación del punto de prueba de paso.
Errores	1. Si el usuario no se encuentra próximo al punto de prueba de paso se devuelve un error. 2. Si el usuario ha enviado erroneamente los datos se devuelve un error. 3. Si el usuario ya ha intentado validar este punto anteriormente se devuelve un error.
Postcondición	–
Notas	

Cuadro 8.7: Consultar puntos de prueba de paso

Consultar puntos de prueba de paso

Descripción	El usuario quiere conocer los puntos de prueba de paso de una ruta.
Precondición	El usuario está logeado en el sistema.
Secuencia	1. El usuario accede a la vista de puntos de prueba de paso. 2. El sistema realiza una búsqueda y muestra los puntos de prueba de paso al usuario.
Errores	–
Postcondición	–
Notas	

Cuadro 8.8: Consultar puntos de recompensa

Consultar puntos de recompensa

Descripción	El usuario quiere conocer los puntos de recompensa.
Precondición	El usuario está logeado en el sistema.
Secuencia	1. El usuario accede a la vista de puntos de recompensa. 2. El sistema realiza una búsqueda y muestra los puntos de recompensa.
Errores	–
Postcondición	–
Notas	

Cuadro 8.9: Consultar cartera

Consultar cartera

Descripción	El usuario quiere conocer el estado de su cartera.
Precondición	El usuario está logeado en el sistema.
Secuencia	1. El usuario accede a la vista de cartera. 2. El sistema busca los datos y los muestra al usuario.
Errores	–
Postcondición	–
Notas	

Cuadro 8.10: Validar recompensa

Validar recompensa

Descripción	El usuario quiere validar una recompensa que ya ha completado.
Precondición	El usuario está logeado en el sistema y ha completado la ruta.
Secuencia	1. El usuario accede a la vista de rutas y muestra al hostelero el código. 2. El hostelero valida con el sistema el código proporcionado por el usuario. 3. El hostelero entrega la recompensa al usuario.
Errores	1. Si el usuario intenta validar una recompensa incompleta o invalida, el hostelero no le entregará el obsequio.
Postcondición	El usuario recibe su recompensa.
Notas	

8.1.3. Hostelero

Cuadro 8.11: Consultar rutas disponibles

Consultar rutas disponibles

Descripción	El usuario quiere conocer las rutas disponibles.
Precondición	El usuario está logeado en el sistema.
Secuencia	1. El usuario accede a la vista de rutas disponibles. 2. El sistema realiza una búsqueda y muestra las rutas disponibles.
Errores	–
Postcondición	–
Notas	

Cuadro 8.12: Consultar puntos de prueba de paso

Consultar puntos de prueba de paso

Descripción	El usuario quiere conocer los puntos de prueba de paso de una ruta.
Precondición	El usuario está logeado en el sistema.
Secuencia	1. El usuario accede a la vista de puntos de prueba de paso. 2. El sistema realiza una búsqueda y muestra los puntos de prueba de paso al usuario.
Errores	–
Postcondición	–
Notas	

Cuadro 8.13: Crear punto de recompensa

Crear punto de recompensa

Descripción	El usuario quiere crear un nuevo punto de recompensa.
Precondición	El usuario está logeado en el sistema y es del tipo hostelero.
Secuencia	1. El usuario accede a la vista para crear un nuevo punto de recompensa. 2. El usuario introduce los datos y envía el formulario. 3. El sistema valida que los datos introducidos sean correctos y crea el nuevo punto de recompensa.
Errores	1. Si alguno de los datos es incorrecto el sistema devuelve un error.
Postcondición	El punto de recompensa está creado en el sistema.
Notas	

Cuadro 8.14: Consultar puntos de recompensa

Consultar puntos de recompensa

Descripción	El usuario quiere conocer los puntos de recompensa.
Precondición	El usuario está logeado en el sistema.
Secuencia	1. El usuario accede a la vista de puntos de recompensa. 2. El sistema realiza una búsqueda y muestra los puntos de recompensa.
Errores	–
Postcondición	–
Notas	

Cuadro 8.15: Actualizar punto de recompensa

Actualizar punto de recompensa

Descripción	El usuario quiere modificar un punto de recompensa.
Precondición	El usuario está logeado en el sistema y es del tipo apropiado.
Secuencia	1. El usuario accede a la vista de puntos de recompensa. 2. El usuario selecciona el punto de recompensa que quiere modificar e introduce los datos a modificar. 3. El sistema valida que los nuevos datos sean correctos y hace las modificaciones.
Errores	1. Si alguno de los datos es incorrecto o no se puede crear por algún motivo se devuelve un error.
Postcondición	El punto de recompensa queda actualizado con los nuevos datos.
Notas	

Cuadro 8.16: Borrar punto de recompensa

Borrar punto de recompensa

Descripción	El usuario quiere eliminar un punto de recompensa.
Precondición	El usuario está logeado en el sistema y es del tipo apropiado.
Secuencia	1. El usuario accede a la vista de puntos de recompensa. 2. El usuario selecciona el punto de recompensa que quiere eliminar. 3. El sistema elimina el punto de recompensa del sistema.
Errores	1. Si el punto de recompensa tiene alguna relación activa en el sistema se devuelve un error.
Postcondición	El punto de recompensa queda eliminado del sistema.
Notas	

Cuadro 8.17: Validar recompensa de un turista

Validar recompensa de un turista

Descripción	El usuario quiere validar la recompensa de un turista.
Precondición	El usuario está logeado en el sistema y es del tipo apropiado.
Secuencia	1. El usuario accede a la vista para validar el punto de recompensa. 2. El usuario escanea el código del turista y envía la petición al servidor. 3. El sistema valida el código.
Errores	1. Si el código no es correcto se devuelve un error. 2. Si el código ya ha sido validado anteriormente se devuelve un error.
Postcondición	El código del usuario queda validado.
Notas	

8.1.4. Institución

Cuadro 8.18: Crear ruta

Crear ruta

Descripción	El usuario quiere crear una nueva ruta.
Precondición	El usuario está logeado en el sistema y es del tipo apropiado.
Secuencia	1. El usuario accede a la vista para crear una ruta. 2. El usuario introduce los datos del formulario y envía la petición al servidor. 3. El sistema valida y crea la ruta.
Errores	1. Si la ruta no es validada devuelve un error.
Postcondición	La ruta queda creada en el sistema.
Notas	

Cuadro 8.19: Consultar rutas

Consultar rutas

Descripción	El usuario quiere consultar las rutas creadas.
Precondición	El usuario está logeado en el sistema.
Secuencia	1. El usuario accede a la vista de las rutas.
Errores	–
Postcondición	–
Notas	

Cuadro 8.20: Actualizar ruta

Actualizar ruta

Descripción	El usuario quiere actualizar una ruta.
Precondición	El usuario está logeado en el sistema y es del tipo apropiado.
Secuencia	1. El usuario accede a la vista de las rutas. 2. El usuario elige la ruta que quiere modificar, rellena los nuevos datos y envia la petición al servidor. 3. El servidor valida y almacena los nuevos datos.
Errores	1. Si la ruta no es validada devuelve un error.
Postcondición	La ruta queda actualizada en el sistema.
Notas	

Cuadro 8.21: Borrar ruta

Borrar ruta

Descripción	El usuario quiere borrar una ruta.
Precondición	El usuario está logeado en el sistema y es del tipo apropiado.
Secuencia	1. El usuario accede a la vista de las rutas. 2. El usuario elige la ruta que quiere borrar. 3. El sistema elimina la ruta del sistema.
Errores	1. Si la ruta no se puede eliminar se devuelve un error.
Postcondición	La ruta queda eliminada del sistema.
Notas	

Cuadro 8.22: Crear punto de prueba de paso

Crear punto de prueba de paso

Descripción	El usuario quiere crear un punto de prueba de paso.
Precondición	El usuario está logeado en el sistema y es del tipo apropiado.
Secuencia	1. El usuario accede a la vista para crear un punto de prueba de paso. 2. El usuario introduce los datos del nuevo punto y envia la petición al servidor. 3. El servidor valida los datos y almacena el nuevo punto.
Errores	1. Si el punto no puede ser creado se devuelve un error.
Postcondición	El punto queda creado en el sistema.
Notas	

Cuadro 8.23: Consultar puntos de prueba de paso

Consultar puntos de prueba de paso

Descripción	El usuario quiere consultar los puntos de prueba de paso.
Precondición	El usuario está logeado en el sistema y es del tipo apropiado.
Secuencia	1. El usuario accede a la vista con el listado de puntos de prueba de paso.
Errores	–
Postcondición	–
Notas	

Cuadro 8.24: Actualizar punto de prueba de paso

Actualizar punto de prueba de paso

Descripción	El usuario quiere actualizar un punto de prueba de paso.
Precondición	El usuario está logeado en el sistema y es del tipo apropiado.
Secuencia	1. El usuario accede a la vista de los puntos de prueba de paso. 2. El usuario elige el punto de prueba de paso que quiere modificar, rellena los nuevos datos y envia la petición al servidor. 3. El servidor valida y almacena los nuevos datos.
Errores	1. Si el punto de prueba de paso no es validado devuelve un error.
Postcondición	El punto de prueba de paso queda actualizado en el sistema.
Notas	

Cuadro 8.25: Borrar punto de prueba de paso

Borrar punto de prueba de paso

Descripción	El usuario quiere borrar un punto de prueba de paso.
Precondición	El usuario está logeado en el sistema y es del tipo apropiado.
Secuencia	1. El usuario accede a la vista de los puntos de prueba de paso. 2. El usuario elige el punto de prueba de paso que quiere borrar. 3. El sistema elimina el punto de prueba de paso del sistema.
Errores	1. Si el punto de prueba de paso no se puede eliminar se devuelve un error.
Postcondición	El punto de prueba de paso queda eliminado del sistema.
Notas	

8.2. Repositorios

Todo el código desarrollado con el prototipo del proyecto se encuentra de forma pública en dos repositorios de GitHub.

- **API** https://github.com/tfg-ucm/api
- **Webapp** https://github.com/tfg-ucm/webapp

8.3. Como desplegar

8.3.1. Preparación del entorno

Para poder desplegar el sistema completo es necesario hacerlo por separado para cada una de las piezas que lo forman. La webapp y la API+almacén son «independientes» entre sí para desplegar ya que es posible que este funcionando la webapp pero la api no esté disponible; no es así en el caso de la API y el almacén de datos, que sí es necesario que se ejecute antes el almacén que la API, pues en su inicio tratará de conectarse para poder realizar el resto de funciones.

El orden propuesto para desplegar es el siguiente:

1. Almacén de datos
2. API
3. WebApp

Almacén de datos

1. Ejecución de mongod con un fichero en concreto `mongod --dbpath ~/mongodb/data/db`

API

1. Instalación de dependencias: `npm install`
2. Ejecución de la API: `npm start`

WebApp

1. Instalación de dependencias: `npm install`
2. Ejecución de la WebApp: `npm start`

8.4. API endpoints

8.4.1. Cuentas

Crear cuenta

Verbo	`POST`
Endpoint	`/api/v1/users`
Descripción	Se crea una nueva cuenta de usuario con los datos recibidos siempre que estos sean correctos.

Consultar cuenta

Verbo	`GET`
Endpoint	`/api/v1/users/:id`
Descripción	Se obtiene la informacion del almacen de datos y se devuelve la cuenta.

Modificar cuenta

Verbo	`PUT`
Endpoint	`/api/v1/users/:id`
Descripción	Con los datos recibidos en la peticion y una vez validados se modifican los existentes en el almacen.

Eliminar cuenta

Verbo	`DELETE`
Endpoint	`/api/v1/users/:id`
Descripción	Si la peticion tiene los permisos necesarios, bien porque un admin quiera eliminar o un usuario quiera eliminar su propia cuenta.

8.4.2. Autenticación

Login

Verbo	`POST`
Endpoint	`/api/v1/users/login`
Descripción	Recibe los datos introducidos por el usuario y los valida con los existentes en la DB. Devuelve el objeto con el token que sera almacenado en el navegador del usuario y con la que ”firmara”las peticiones que haga en el futuro a la API.

Logout

Verbo	`POST`
Endpoint	`/api/v1/users/logout`
Descripción	Elimina la sesion del almacen de datos.

8.4.3. Puntos de prueba de paso

Crear punto

Verbo	`POST`
Endpoint	`/api/v1/beacons`
Descripción	Se crea un nuevo punto de prueba de paso con los datos necesarios del beacon y su localizacion siempre que estos sean correctos.

Consultar punto

Verbo	`GET`
Endpoint	`/api/v1/beacons/:id`
Descripción	Se obtiene la informacion del almacen de datos y se devuelve un json.

Listado de puntos

Verbo	`GET`
Endpoint	`/api/v1/beacons`
Descripción	Obtiene un listado con los puntos registrados por una cuenta.

Modificar punto

Verbo	`PUT`
Endpoint	`/api/v1/beacons/:id`
Descripción	Con los datos recibidos en la peticion y una vez validados se modifican los existentes en el almacen.

Eliminar punto

Verbo	`DELETE`
Endpoint	`/api/v1/beacons/:id`
Descripción	Si la peticion tiene los permisos necesarios el punto de prueba de paso se elimina.

Validar punto

Verbo	`POST`
Endpoint	`/api/v1/beacons/validate`
Descripción	El sistema valida un punto de prueba de paso para un usuario.

8.4.4. Puntos de recompensa

Crear punto de recompensa

Verbo	`POST`
Endpoint	`/api/v1/rewards/point`
Descripción	Se crea un nuevo punto de recompensa con los datos necesarios y su localizacion siempre que estos sean correctos.

Consultar punto de recompensa

Verbo	`GET`
Endpoint	`/api/v1/rewards/point/:id`
Descripción	Se obtiene la informacion del almacen de datos y se devuelve un json.

Listado de puntos de recompensa

Verbo	`GET`
Endpoint	`/api/v1/rewards/point`
Descripción	Se obtiene un listado con los puntos de recompensa creados por una cuenta.

Modificar punto de recompensa

Verbo	`PUT`
Endpoint	`/api/v1/rewards/point/:id`
Descripción	Con los datos recibidos en la peticion y una vez validados se modifican los existentes en el almacen.

Eliminar punto de recompensa

Verbo	`DELETE`
Endpoint	`/api/v1/rewards/point/:id`
Descripción	Si la peticion tiene los permisos necesarios el punto de recompensa se elimina.

8.4.5. Recompensas

Crear recompensa

Verbo	`POST`
Endpoint	`/api/v1/rewards`
Descripción	Al crear una recompensa se especifican las pruebas de paso necesarias y los puntos de recompensa autorizados a otorgarlas.

Consultar recompensa

Verbo	`GET`
Endpoint	`/api/v1/rewards/:id`
Descripción	Se obtiene la informacion del almacen de datos y se devuelve un json.

Listado de recompensas

Verbo	`GET`
Endpoint	`/api/v1/rewards`
Descripción	Obtiene un listado con las recompensas que ha recibido un usuario.

Modificar recompensa

Verbo	`PUT`
Endpoint	`/api/v1/rewards/:id`
Descripción	Con los datos recibidos en la peticion y una vez validados se modifican los existentes en el almacen.

Eliminar recompensa

Verbo	`DELETE`
Endpoint	`/v1/rewards/:id`
Descripción	Si la peticion tiene los permisos necesarios la recompensa se marca como ”no entregable”.

Validar recompensa

Verbo	`POST`
Endpoint	`/v1/rewards/validate`
Descripción	Valida que todos los puntos esten validados y emite un pago al turista.

Reclamar recompensa

Verbo	`POST`
Endpoint	`/v1/rewards/claim/:name`
Descripción	Realiza el pago al establecimiento para recibir el obsequio final.

Referencias

1. *Mastering Bitcoin* https://github.com/bitcoinbook/bitcoinbook
2. *Stellar for developers* https://www.stellar.org/developers/
3. *Ethereum for developers* https://www.ethereum.org/developers/
4. *NodeJS documentation* https://nodejs.org/en/docs/
5. *NPM documentation* https://docs.npmjs.com/
6. *ExpressJS documentation* https://expressjs.com/en/api.html
7. *MongoDB documentation* https://docs.mongodb.com/manual/
8. *Mongoose documentation* https://mongoosejs.com/docs/api.html
9. *ECMAScript 6* https://www.w3schools.com/js/js_es6.asp
10. *Agile Project Management* https://www.atlassian.com/agile/project-management
11. *WebStorm documentation* https://www.jetbrains.com/help/webstorm/documentation.html
12. *Visual Studio Code REST Client* https://marketplace.visualstudio.com/items?itemName=humao.rest-client
13. *TEXIS* http://gaia.fdi.ucm.es/research/texis/